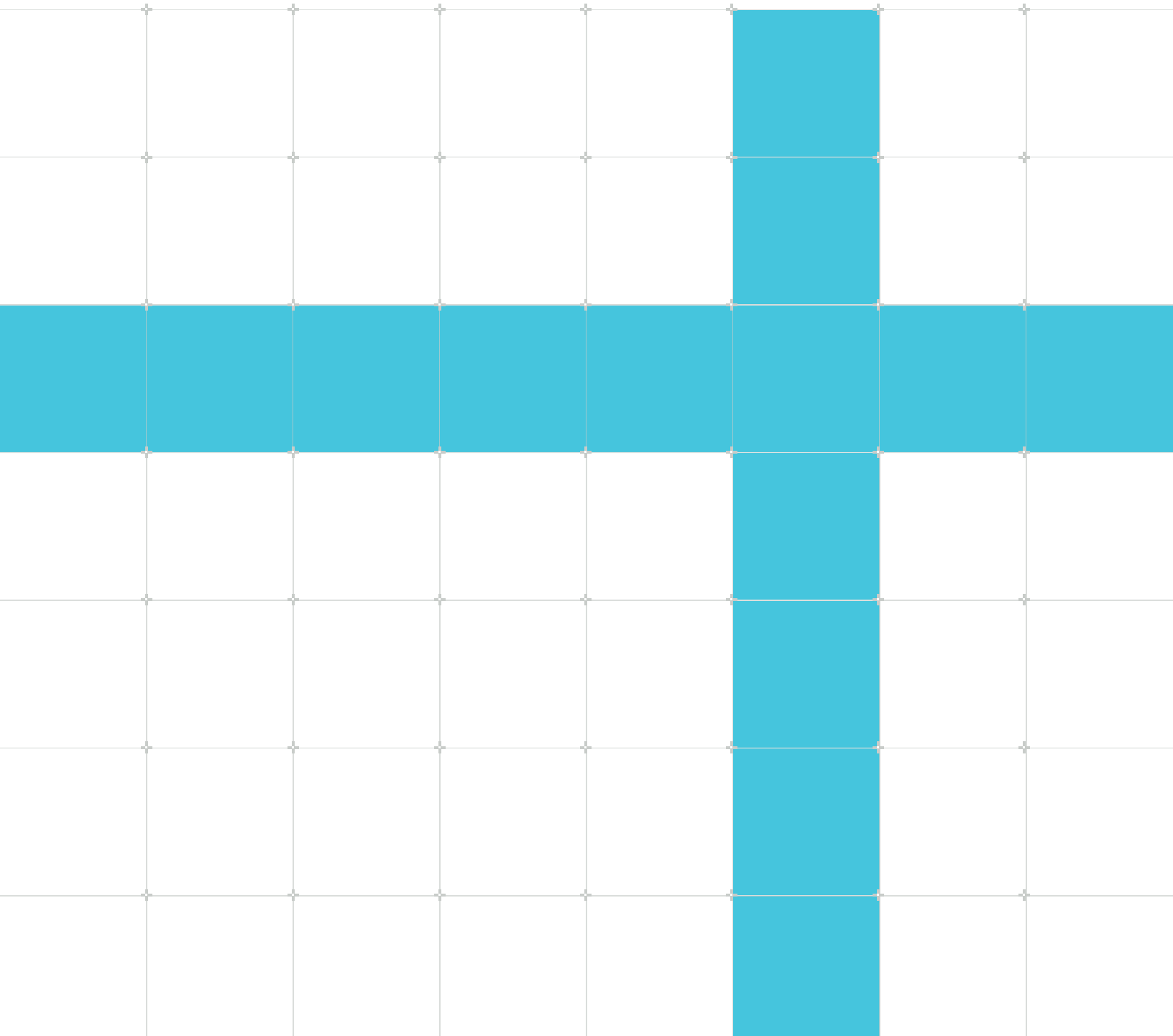
Arm® CoreLink™ DMA-250 Controller

Revision r0p0

Technical Reference Manual

Non-Confidential

Issue 02
108001_0000_02_en



Arm® CoreLink™ DMA-250 Controller Technical Reference Manual

This document (108001_0000_02_en) was issued on 2024-09-20. There might be a later issue at http://developer.arm.com/documentation/108001

The product revision is r0p0.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses the CoreLink™ Direct Memory Access DMA-250 Controller.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com.

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Contents

1. DMA-250 overview

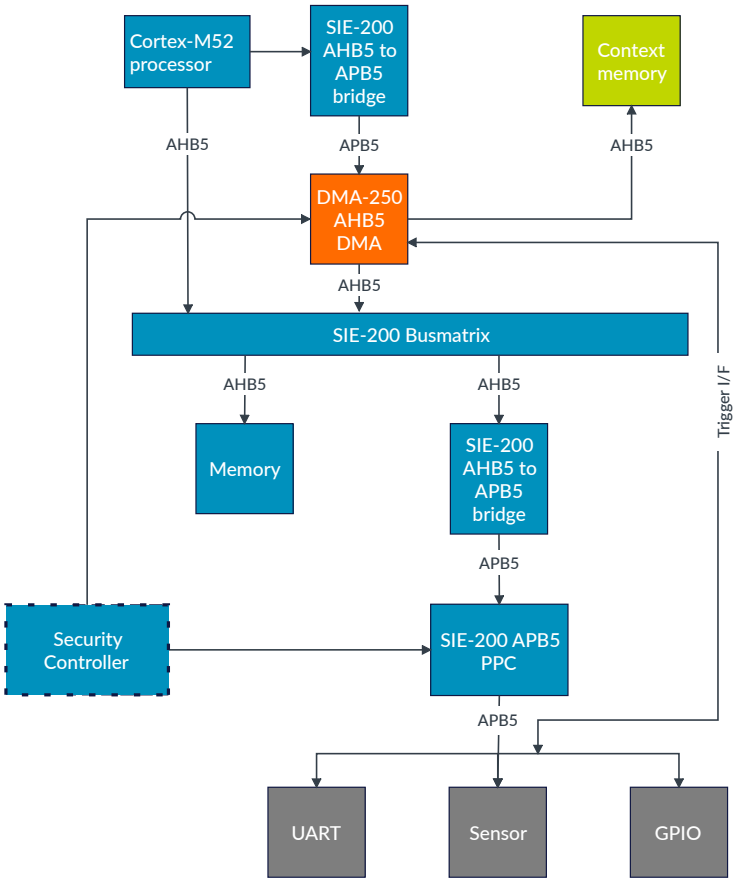
CoreLink DMA-250 is a Direct Memory Access Controller (DMAC) with AMBA® AHB5 interfaces, which provides fast memory to memory, peripheral to memory, memory to peripheral copy, and peripheral to peripheral capabilities on multiple channels.

The processor in the system can control the DMA channel behavior over an APB5 interface and set security-related settings when Security Extension is supported. The DMAC has configurations for different types of copy, scatter-gather, and increment. It also supports trigger inputs and outputs for flow control and command sequencing capabilities. DMA-250 adds support for low-power integration through the LPI interfaces for both clock and power.

DMA-250 structurally divides DMA channels to virtual channels (VCH) and physical channels (PCH). Virtual channels implement the aspects that are unique to each DMA channel while physical channels consist of logic and resources that are identical for each DMA channel. DMA-250 implements only a few physical channels which it shares between multiple virtual channels.

The following figure shows DMA-250 in a system, where optional elements are shown with dashed lines:

Figure 1-1: DMA-250 in a system



Key features

DMA-250 supports the following key features:

- AHB5 data path
- Dedicated AHB5 interface to access virtual channel context area
- APB5 (Issue D) configuration registers
- 1, 2, or 4 parallel physical DMA channels and 2 or 4 virtual channels per physical channel
- 1D memory copy including increments
- Automatic command restart and register reload
- Command linking
- Automatic booting support
- Interrupt capability for each channel and global events
- Optional TrustZone® support and security settings per channel. For details, see *TrustZone® technology for Armv8-M Architecture*
- Optional trigger input and output ports

- Optional General-Purpose Output (GPO) per channel
- Cross Trigger Interface (CTI) for debug purposes
- LPI Q-Channel and P-Channel interfaces for clock quiescence and power control, respectively

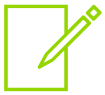
1.1 Compliance

DMA-250 interfaces are compliant with the following Arm specifications and protocols.

- AMBA APB5 protocol, see the *AMBA® APB Protocol Specification*
- AMBA Low Power Interface protocol, see the *AMBA® Low Power Interface Specification*
- AMBA AHB5 protocol , see the *AMBA® 5 AHB Protocol Specification*

1.2 Configurable RTL options

DMA-250 can be configured with the following RTL options to meet specific design requirements.


Note

The parentheses contain the relevant configuration parameter. These are described in the *Arm® CoreLink™ DMA-250 Controller Configuration and Integration Manual*.

- Channel identification register configurable width between 0-bit and 16-bit (CHID_WIDTH).
- General Purpose Output (GPO) configurable width of the channels between 0-bit and 32-bit (GPO_WIDTH).
- FIFO depth of all channels to 1, 2, or 4 (FIFO_DEPTH).
- Data width of the bus interface 32 or 64 (DATA_WIDTH).
- The number of configurable DMA physical channels is 1, 2, or 4 (NUM_PCH)
- The number of virtual channels (NUM_VCH). This is calculated using the ratio of virtual channels per physical channel (VCH_PCH_RATIO = 2, 4, or 8). So NUM_VCH = NUM_PCH * VCH_PCH_RATIO.
- Presence of Security Extension for TrustZone® support (SECEXT_PRESENT).
- Supports configurable triggers (HAS_TRIG). When configurable triggers are enabled (HAS_TRIG = 1), two additional RTL options are supported:
  - A bitmask for enabling the synchronizers on the trigger input interfaces (TRIG_IN_SYNC_EN_MASK).
  - A bitmask for enabling the synchronizers on the trigger output interfaces (TRIG_OUT_SYNC_EN_MASK).

1.3 Product documentation and design flow

This section describes the DMA-250 documentation in relation to the design flow.

1.3.1 Documentation

The DMA-250 documentation is as follows:

Technical Reference Manual

The Technical Reference Manual (TRM) describes the functionality and the effects of functional options on the behavior of DMA-250. It is required at all stages of the design flow. The choices made in the design flow can mean that some behaviors described in the TRM are not relevant.

Configuration and Integration Manual

The Configuration and Integration Manual (CIM) describes:

- The available build configuration options and related issues in selecting them.
- How to integrate DMA-250 into an SoC. This section describes the considerations need to be taken into account by the integrator when connecting the interfaces in the system.
- The processes to sign off on the configuration, and integration of the design.

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

Implementation Base Enablement Package User Guide

The implementation Base Enablement Package (iBEP) provides guidance to start the physical implementation process, with specific inputs and guidelines, constraints, and floorplanning information.

If you are programming DMA-250, contact the implementer to determine:

- The build configuration of the implementation.
- The integration, if any, that was performed before implementing DMA-250.
- The integrator to determine the pin configuration of the device that you are using.

1.3.2 Design flow

Arm delivers DMA-250 as synthesizable RTL.

Before you can use it in a product, it must go through the following processes:

Implementation

The implementer configures and synthesizes the RTL to produce a soft netlist.

Integration

The integrator connects the implemented design into an SoC. Integration includes connecting the design to a memory system and peripherals.

Programming

The system programmer develops the software to configure and initialize DMA-250, and tests the required application software.

Each process is separate and can include implementation and integration choices that affect the behavior and features of DMA-250.

The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed.

These options usually include or exclude logic that affects one or more of the following:

- Area
- Maximum frequency
- Features of the resulting macrocell

Configuration inputs

The integrator configures some features of DMA-250 by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made.

Software configuration

The programmer configures DMA-250 by programming particular values into registers. This configuration affects the behavior of DMA-250.

1.4 Product revisions

This section describes the differences in functionality between product revisions:

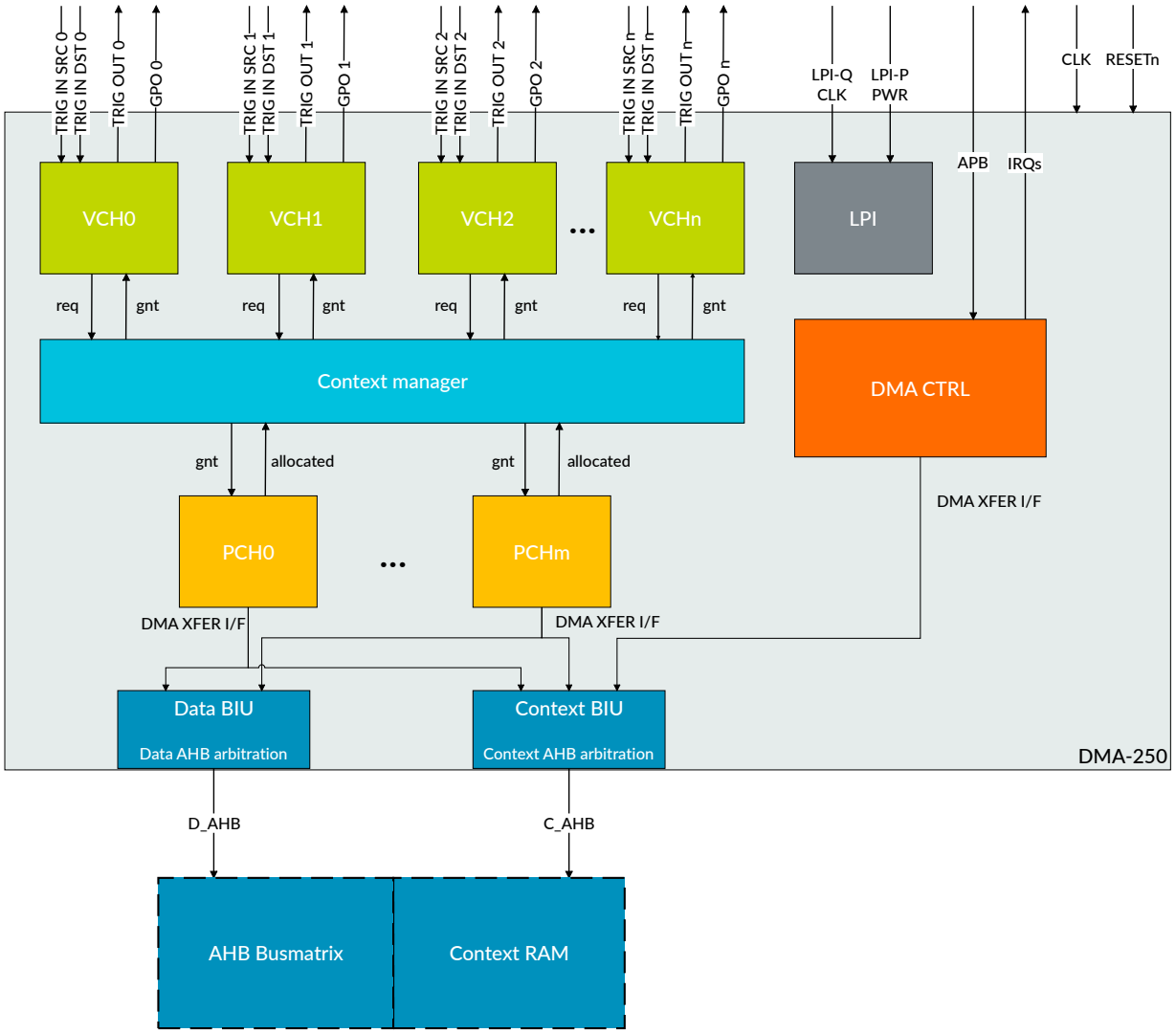
r0p0 First release.

2. Component overview

The internal structure of DMA-250 is split into blocks that have their own specific functions.

The following block diagram shows the main blocks of DMA-250:

Figure 2-1: DMA-250 top-level block diagram



DMA-250 has the following main blocks:

DMA CTRL

Control Block

The control block on DMA unit level can define additional settings for each channel and select which security world the channels belong to. It also collects global status information and interrupts from the virtual channels for easier management of the whole DMA.

Implements common DMA registers, APB interface, register clear feature, and CTI.

For details, see:

- Cross Trigger Interface
- Programmers model
- Register clear feature
- APB5 subordinate interface

VCH

DMA Virtual Channel Block

- Implements one functional unit of the DMA, which is capable of managing a DMA job.
- Several Virtual channels (VCHs) can exist, this is determined by calculated parameter NUM_VCH.

For details, see Virtual channel.

Context manager

- Arbitrates between allocation requests from VCHs.
- Implements the down-stream multiplexers between VCHs, up-stream multiplexers between PCHs.

For details, see Context manager.

PCH

DMA Physical Channel Block

- Implements one low-level execution unit of the DMA, that is capable of DMA transfers.
- Several PCHs may exist determined by configuration parameter NUM_PCH.

For details, see Physical channel.

BIU

Data/Context Bus Interface Unit

The Data BIU arbitrates DMA data transfer requests or command descriptor read requests during command linking, whereas the Context BIU is used for accessing the Context Memory area.

The Data BIU contains the single Data FIFO which is shared between the PCHs.

For details, see Bus Interface Unit.

QCTRL

Power and Clock management Implements the clock and power management features of DMA-250.

For details, see:

- DMAC Power management

- LPI interfaces

2.1 Virtual channel

Virtual channels (VCH) implement the aspects that are unique to each DMA channel, while physical channels consist of logic and resources that are identical for each DMA channel. The DMA architecture supports more virtual channels than physical ones implemented in the DMA. This makes it available for the software to configure channels virtually in the register bank that will compete for the physical resources.

Each VCH has the following attributes:

- Channel control and pause Finite-State Machines (FSM) that track and coordinate the DMA command's lifecycle tha the channel is executing.

  The control FSM manages the lifecycle of the channel. It executes:

  - ENABLE, DISABLE, PAUSE, RESUME, STOP commands.
  - Validate command configuration.
  - Wait for trigger inputs.
  - Wait for allocation.
  - Handle being in preemptive paused state.
  - Perform automatic command restart and command link fetch.
- Registers that need to be implemented locally. see DMA Channel Register Frame.
- Trigger interfaces and the related control logic when trigger support is configured.

  The DMA channels can have peripheral trigger ports for autonomous command and flow control between the DMA and the peripherals. Each virtual channel has its own trigger control block which handles the corresponding external trigger interfaces.
- GPO interface when GPO support is configured.

A VCH cannot perform DMA transfers without being allocated to a PCH. Each VCH tracks its own allocation status with their context FSM logic, which handles the interactions with the Context manager module. The Context manager controls the virtual channel allocation/ deallocation to physical channels.

The number of virtual channels is calculated as the multiple of NUM_PCH * VCH_PCH_RATIO. That is the number of physical channels multiplied by the number of virtual channels per physical channel.

Allocation of a virtual channel

The virtual channel requests allocation when the virtual channel control FSM is in one of the following states.

- Disabled, Done, and Done paused states when the channel is ready to send transfers (for example: channel has been enabled, resumed, command linking disabled, triggers disabled).

- Command trigger state if trigger request is HIGH.
- Flow control trigger state if all configured trigger requests are HIGH.
- Done and Done paused states when the channel is ready to fetch the next command descriptor (that is command link is enabled, channel is resumed, and command restart disabled).

Related information

- Physical channel
- Context manager
- Allocation priority

2.1.1 Allocation priority

Allocation arbitration between virtual channels is done based on their priority.

The highest priority level virtual channel wins the arbitration. If two or more virtual channels have the same priority, arbitration between virtual channels is done based on the least recently granted scheme. Based on the least recently granted scheme the oldest selected virtual channel wins the arbitration.

2.2 Physical channel

The physical channels (PCH) implement the read and write control logic, including the burst calculation logic and pipelines. Each physical channel is allocated to one virtual channel at a time.

Physical channels also implement the command linking logic for reading command descriptors and based on those performing the necessary updates to the relevant registers in the PCH/VCH. Each physical channel implements its own context fetch or save logic, as well as the register reload logic which is used by commands for which automatic reload is set.

The physical channels are connected to the bus interface units (BIU) which serve as arbiters between physical channels that operate in parallel and generate transfers on their AHB5 interface as requested by the PCHs.

Related information

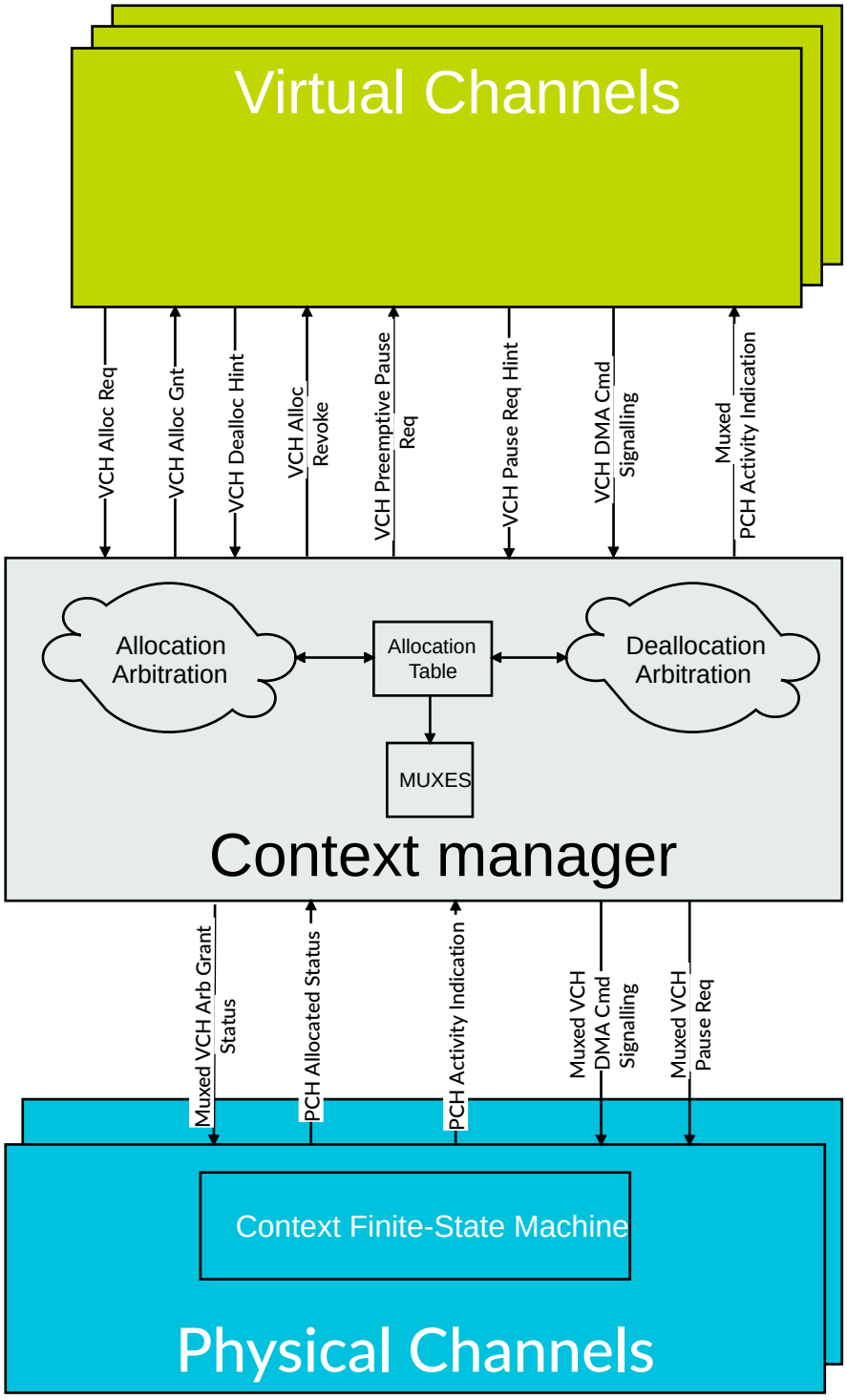
- Virtual channel
- Context manager
- Allocation priority
- Bus Interface Unit

2.3 Context manager

The Context manager provides the arbitration layer between the virtual channels and the physical channels. It decides which virtual channel (VCH) gets allocated to which physical channel, and if necessary, which virtual channel gets deallocated.

By granting or revoking access, the Context manager triggers the relevant processes in the involved VCH and PCH that will result in their allocation or deallocation, respectively. The multiplexing of any signalling between the VCH (virtual channel) and PCH (physical channel) is also handled by the Context manager.

Figure 2-2: Context Manager High Level Block Diagram



Allocation conditions

After reset the DMA automatically allocates virtual channels to all physical channels. The allocation is based on the corresponding virtual and physical channel IDs. For example, VCH0 to PCH0, VCH1 to PCH1 and so on.

When a deallocated VCH is in a ready-to-run state, it requests allocation from the Context manager. This happens when all of the following is true:

- VCH is enabled.
- If triggers are used, then all configured trigger requests have arrived.
- No pause request for the given VCH.

Deallocation conditions

When VCHs reach the point in their operation when they do not need the functionality of a physical channel anymore (even temporarily), they give a hint for deallocation to the Context manager, which arbitrates between these hints and selects a VCH to be revoked.

Allocated VCHs signal a deallocation hint when any of the following is true:

- VCH is disabled (reached the end of the job or stopped).
- VCH is waiting for a trigger request.
- VCH is paused.

Related information

- Separate AHB5 ports for data and virtual channel context
- Virtual channel
- Physical channel
- Channel arbitration for AHB5
- Allocation priority
- Context memory area size
- Context switch processes
- Allocation and deallocation arbitration

2.3.1 Arbitration policy

The Data BIU and the Context BIU use the same arbitration policy.

The arbitration policy is separated to two layers depending on the priority setting of the channel:

- Priority-based arbitration layer that arbitrates between channels of different priority levels.
- Round-robin scheme arbitration layer that arbitrates between channels with the same priority level, based on which of them was least recently granted.

The base layer is a fixed arbitration scheme if the priority settings are different per channel:

- When multiple requests have different priorities, the single request with the highest priority wins the arbitration.
- When multiple channels have the highest priority a round robin scheme, least-recently granted (LRG) is applied.
- When all channels have the same priority then all channels are using the round-robin arbitration scheme at every arbitration point.
- The requestor priorities of the register clear logic and the APB5 to Context memory bridge logic are handled dynamically so that they are always match the priority level of the highest priority channel in the arbitration.
- When one channel must have higher priority than the others, it can be escalated to the fixed priority layer by setting the priority value above the others. This can lead to starvation of low-priority requests but it ensures a deterministic execution of high priority tasks.

A channel using the fixed priority scheme has a higher priority than the round robin scheme so it gets arbitrated first until it runs out of requests. This scheme keeps the flexibility for the software to set any kind of priority policy in the system. This also allows for time-critical tasks to happen and channels with the same priority use the bus in an even share.

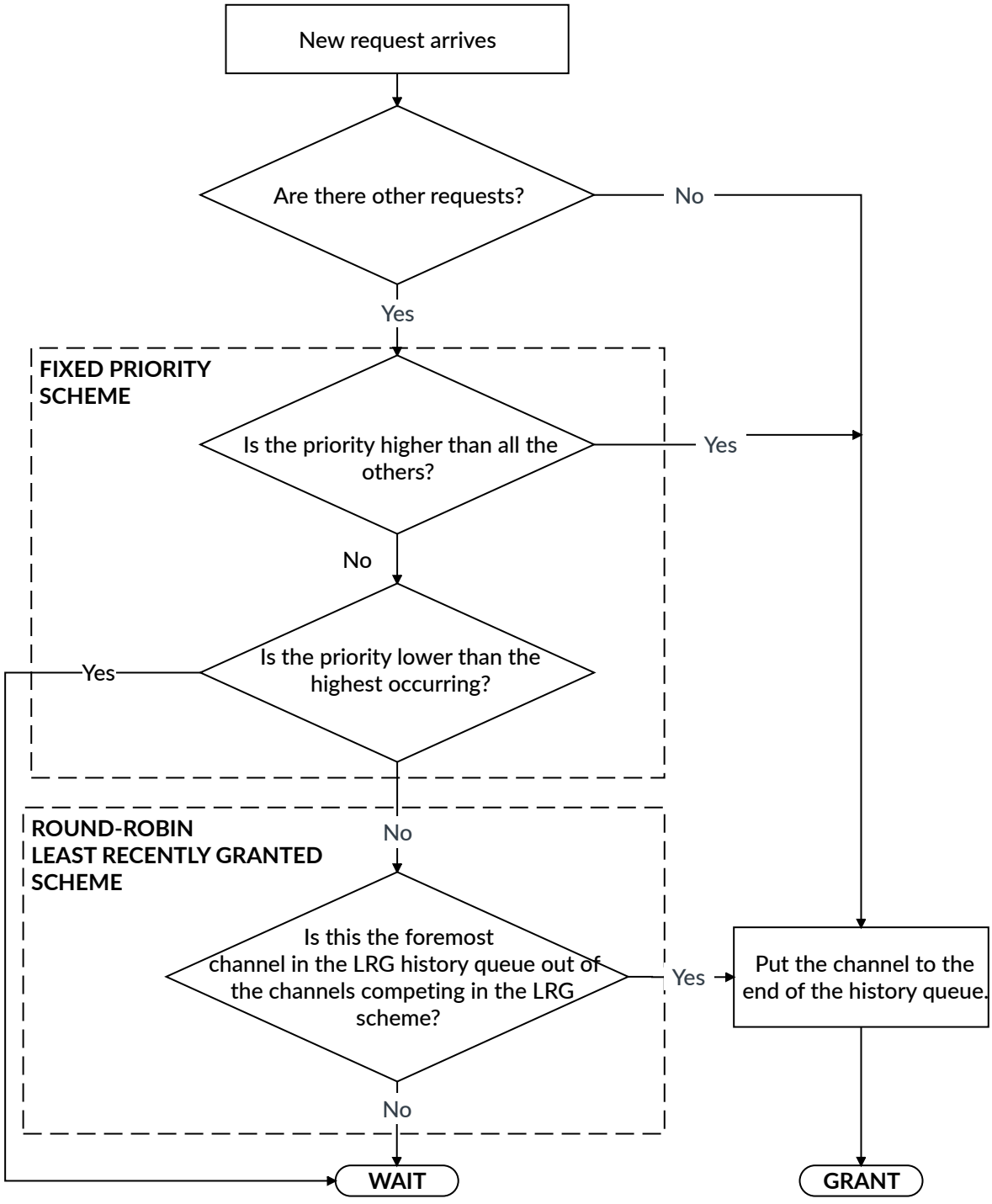
The round robin scheme saves the history of the granted requests. The scheme sorts the requests by descending order of the time of grant, that is from the least recently granted to the most recently granted.

When multiple requests have higher priority, the round robin scheme only considers the requests with the highest priority level and selects the least recently granted one based on the serving history. The same applies when all requests have been served on the higher priority level and the same channels have new requests on a lower priority level which has more outstanding requests.

That is, the round robin scheme considers all the requests with the same (highest) priority level and selects the least recently granted channel.

The following figure describes the LRG algorithm in detail:

Figure 2-3: LRG arbitration algorithm



2.3.2 Allocation and deallocation arbitration

When deciding allocation, the Context manager applies the arbitration policy based on the CH_CTRL.CHPRIO register field values of the requesting VCHs.

For details of the Arbitration policy, see Arbitration policy.

When deciding deallocation, the Context manager bases its allocation policy on CH_CTRL.CHPRIO register field values in an inverse manner so that lower priority channels are selected for deallocation. The Context manager also considers the status of the channels with asserted deallocation hints and prefers disabled channels (regardless of their priority) over channels that are enabled but ARE waiting for a trigger event or to become un-paused.

Channels that are actively performing DMA transfers might take a long time before they assert their deallocation hints, which might result in long waitng times before deallocation arbitration can occur. For example, when a channel is transferring a large amount of data at a time, or if the channel does not get arbitrated on the Data AHB for a long time due to other, higher priority channels winning the bus arbitration.

The Context manager implements a preemptive pause mechanism that results in earlier deallocation arbitration if necessary. This ensures that a VCH with a higher priority job can switch out a lower priority VCH and can quickly get allocation, instead of being stalled due to the lower priority VCH not getting deallocated.

For more details, see Preemptive pause feature.

2.3.3 Context switch processes

Allocation and deallocation decisions made by the Context manager result in Context AHB traffic to load/store configuration register values from/to the Context memory.

Context switches results in latency- and bandwidth- overhead, which depends on the VCH-to-PCH ratio in the selected render configuration and the number of VCHs running in parallel.

Channel priorities affect allocation and deallocation arbitration and thus context switching occurrences, as well as AHB5 bus arbitration. For details, see Channel arbitration for AHB5.

DMA-250 does not implement any fair share quality of service algorithms; therefore software must set the channel priority levels of the virtual channels properly by configuring CHCTL.CHPRIO register field. If there are no distinguishing factors to a DMA job such as time criticality, Arm recommends using the same priority level for all running channels.


Note

The context handling is fully done by HW, the internal operation and the context status of a VCH is not visible to the software.

2.3.4 CHLOCKREQ feature

CH_CTRL.CHLOCKREQ is a special register field for disabling deallocation of a given VCH.

If the CH_CTRL.CHLOCKREQ bit is set, then after the VCH is allocated, it does not lose the allocation even if it finishes its job (reaches disabled state).

The purpose of this feature is to minimize the allocation overhead for timing sensitive DMA jobs. The D-AHB bus arbitration is still performed based on the CHPRIO value, so a low priority VCH can be allocated due to the CH_CTRL.CHLOCKREQ bit. However, it might not reach the D-AHB bus if there are other higher priority VCHs allocated and generating data transfers.

To avoid starvation issues due to the CHLOCKREQ feature, if this field is set for all the allocated channels, then this feature is disabled. Therefore, if NUM_PCH =1, this CHLOCKREQ feature is not supported.

2.3.5 Preemptive pause feature

Preemptive pause prevents priority inversion in the DMA, which is when a high priority virtual channel cannot access a physical channel due to a lower priority virtual channel being unable to progress its operation. As it cannot progress, it never reaches a point where it could give up its shared resources to the high priority virtual channel.

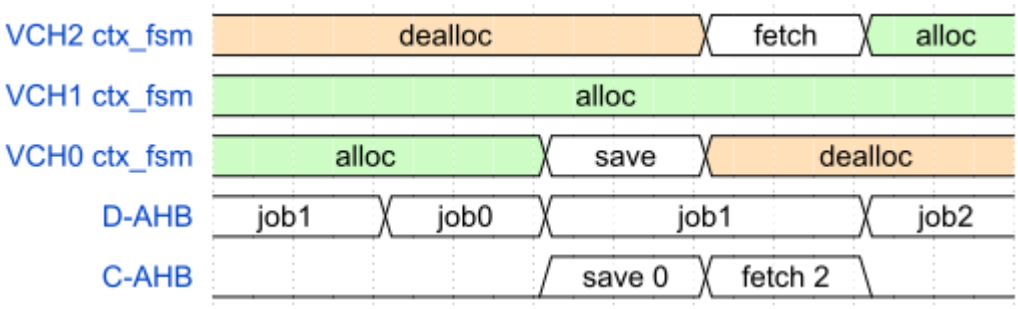
The preemptive pause request is issued by the context manager when it detects an incoming allocation request from a high priority channel that cannot be granted for the reason described above. The preemptive pause request is issued to the lowest priority allocated virtual channel.

Example scenario:

There are 3 VCHs enabled with different priorities: low (VCH0), medium (VCH1), high (VCH2).

- T2: VCH0 and VCH1 are already allocated and running, when VCH2 was enabled.
- T3: Once VCH2 is started, the low priority VCH0 receives a preemptive pause request and will progress with its job so it reaches a state where it can give up its allocation as fast as possible.
- T5: VCH0 reached a state where it can be deallocated, it is paused.
- T7: VCH2 starts the context fetch.
- T9: VCH2 is allocated and starts generating data transfers.

Figure 2-4: Preemptive pause feature



Context processes

There are different type of transfers targeting the Context memory:

- Context save/fetch.
- Register clear for deallocated channels (REGCLR). For more details, see Register clear feature.
- APB accesses bridged to C-AHB.

There may be a race condition between these processes. As a general rule, the order of the different C-AHB transactions is in the order that they were initiated, this ensures coherency.

Some examples:

APB transfers getting extended with wait states due to context switch

If there is an ongoing context fetch/save while an APB write targets the same channel's DMACH register frame, then first the fetch/save operation will be finished, and only after that will the APB access proceed.

Delayed context switch

If there is an ongoing APB write transaction to CNTXBASE, SECCFG registers, then the context switch processes must wait until the APB access is done.

APB transfers getting extended with wait states due to REGCLR

If there is a pending register clear request (for example change of channel security, or CNTXBASE address), then an incoming APB write transaction to the DMACH register frame must wait until the register clear process is complete.

Otherwise, if the processes are exchanged, then the latter APB write data would be lost. The register clear process might take a considerable amount of time if there are pending register clear requests for many channels.

2.4 Context memory

The context memory is an external memory where DMA-250 stores the context of each channel.

A channel's context refers to the architectural and microarchitectural information based on which the channel operates and performs DMA transfers. A part of this context is stored within the DMA

(in the VCH register bank). The rest is stored in the context memory from where it is loaded into a PCH (context fetching) to which the VCH gets allocated, or written back (context saving) when the VCH gets deallocated.

2.4.1 Context memory area size

The Context Memory area is a continuous area that is private to the DMAC. The DMAC accesses it through the Context AHB5 interface.

The size of the required Context Memory (in bytes) can be calculated with the following formula: CTX_MEM_SIZE = NUM_VCH * 64 * (1 + SECEXT_PRESENT) bytes.

For example:

- For a 4x VCH configuration without TrustZone support (NUM_VCH = 4 and SECEXT_PRESENT = 0), the necessary Context Memory area is 256 bytes.
- For a 8x VCH configuration with TrustZone support (NUM_VCH = 8, SECEXT_PRESENT = 1), the necessary Context Memory area is 1024 bytes.

2.5 Bus Interface Unit

The main functionality of the Bus Interface Unit (BIU) is to arbitrate between multiple entities that request AHB5 transfers on the given BIU's AHB5 interface, and to generate the requested AHB5 accesses that won the arbitration.

The DMA has one BIU for the Data AHB5 interface and one BIU for the Context AHB5:

- The Data BIU arbitrates between multiple PCHs when they request DMA transfers or command link reads on the Data AHB5. The Data BIU implements the only FIFO that is implemented in DMA-250.
- The Context BIU arbitrates between multiple PCHs, the register clear logic, and the APB5 to Context memory bridge logic when they request transfers towards the Context memory on the Context AHB5.

Related information

- Channel arbitration for AHB5
- Separate AHB5 ports for data and virtual channel context

3. DMAC interfaces

This section contains an overview of the CoreLink DMA-250 interfaces.

3.1 Clock and reset interface

DMA-250 is designed to be in a single clock and reset domain.

DMA-250 has a common clock and reset input. The APB5 completer interface can be clock gated externally using its pclken signal, which enables middle level clock gating outside the DMA.

There are no special power up requirements defined for DMA-250.

Table 3-1: Clock and reset interfaces

Signal name	Direction	Connection	Description
clk	Input	Top	The single clock input of the DMA Controller.
resetn	Input	Top	Active-LOW reset. This reset must be deasserted synchronously with the clk clock signal, but it can be asserted asynchronously.

3.2 APB5 subordinate interface

The APB5 subordinate interface enables access to the internal configuration registers of DMA-250.

These include:

- DMA Channel configuration
- DMA Security configuration
- DMA Secure and Non-Secure controls configuration
- Access (Read only) to the DMA configuration

Only 32-bit wide accesses are supported.

An error response is returned:

- When an APB5 Write-Access uses pstrb other than `0b1111`
- A register access security violation occurs, and the software configurable register setting selects the error response.

Unaligned accesses are treated as normal accesses by having paddr[1:0] tied to 0 internally.

Only the PWAKEUP function that is added by APB5 is implemented in DMA-250.

For more information, see the *AMBA® APB Protocol Specification* document.

Related information

- Memory map
- APB5 access routing
- Separate AHB5 ports for data and virtual channel context
- APB5 signals

3.2.1 APB5 protection

Certain registers in the APB5 register space are protected for higher privilege and security software modules. These registers configure the overall behavior of the DMAC and are protected internally within the DMAC by checking the privilege and security attribute of the access.

Privilege protection

The DMAC checks pprot[0] bit when accessing registers with higher privilege rights. The DMAC returns with **RAZ/WI** when the access is denied.

Security protection

The DMAC supports the Security Extension by checking pprot[1] when accessing registers with higher security rights. When accesses are violating security the DMAC returns **RAZ/WI** or error response depending on the software configurable register and it might also generate an interrupt if enabled.

3.2.2 APB5 pwakeup signal

The APB5 interface is extended with a clock wakeup signaling through the pwakeup port. This allows faster clock request mechanism when the pwakeup is generated from a registered source.

This signal is used to indicate that there is ongoing activity that is associated with the APB5 interface. If pwakeup is not implemented on the signals driving the APB5 interface, then the psel signal can be registered to generate the pwakeup signal.

Related information

- APB5 signals

3.2.3 APB5 pdebug signal

The APB5 interface has a sideband signal pdebug that allows a debugger to access the DMAC without causing side-effects when sweeping the memory range in Non-secure mode. When the debugger creates the APB5 access this signal can be set HIGH, which disables the security

violation error and interrupt generation while still protecting the Secure registers by responding with **RAZ/WI**. This signal must be stable throughout the 2 cycles of the access.

Related information

- APB5 signals

3.3 AHB5 manager interfaces

DMA-250 has two AHB5 interfaces: one for the DMA transfers, and another dedicated to the DMAC microarchitectural context memory. On both interfaces, the DMA is an AHB5 manager with a reduced set of AHB5 features.

Table 3-2: Supported AHB5 features

AHB5 bus properties	Supported AHB5 features
Extended_Memory_Type	DMA-250 implements full HPROT signaling
Secure_Transfers	When TrustZone support is configured (SECEXT_PRESENT), DMA-250 supports the concept of Secure and Non-secure transfers and implements HNONSEC signal
Write_Strobes	DMA-250 implements HWSTRB but does not send sparse writes.
Single-copy atomicity size	The size of a single-copy atomic group depends on the transfer size: • Minimum: 8-bit • Maximum: 32-bit
User signaling	DMA-250 supports configurable user signals.

Related information

- Table A-5: DATA AHB5 signals on page 167
- Table A-6: CONTEXT AHB5 signals on page 168

3.3.1 Separate AHB5 ports for data and virtual channel context

DMA-250 stores the virtual channel context data in a memory outside of the IP. The main anticipated use case is that this context memory is the DMA's fixed, private area in the system SRAM.

However, the performance of DMA-250 is increased if its DMA accesses and context memory accesses do not compete with each other for the same AHB5 interface. For this reason, DMA-250 has two separate AHB5 interfaces:

- Data AHB5 interface dedicated to DMA transfers
- Context AHB5 interface dedicated to context memory accesses

This way, DMA-250 can be connected either to a full crossbar multilayer AHB5 busmatrix, or to a dedicated SRAM or register file that is used as channel context storage.

The following figures shows the two possible arrangements.

Figure 3-1: Separate SRAM dedicated to context data

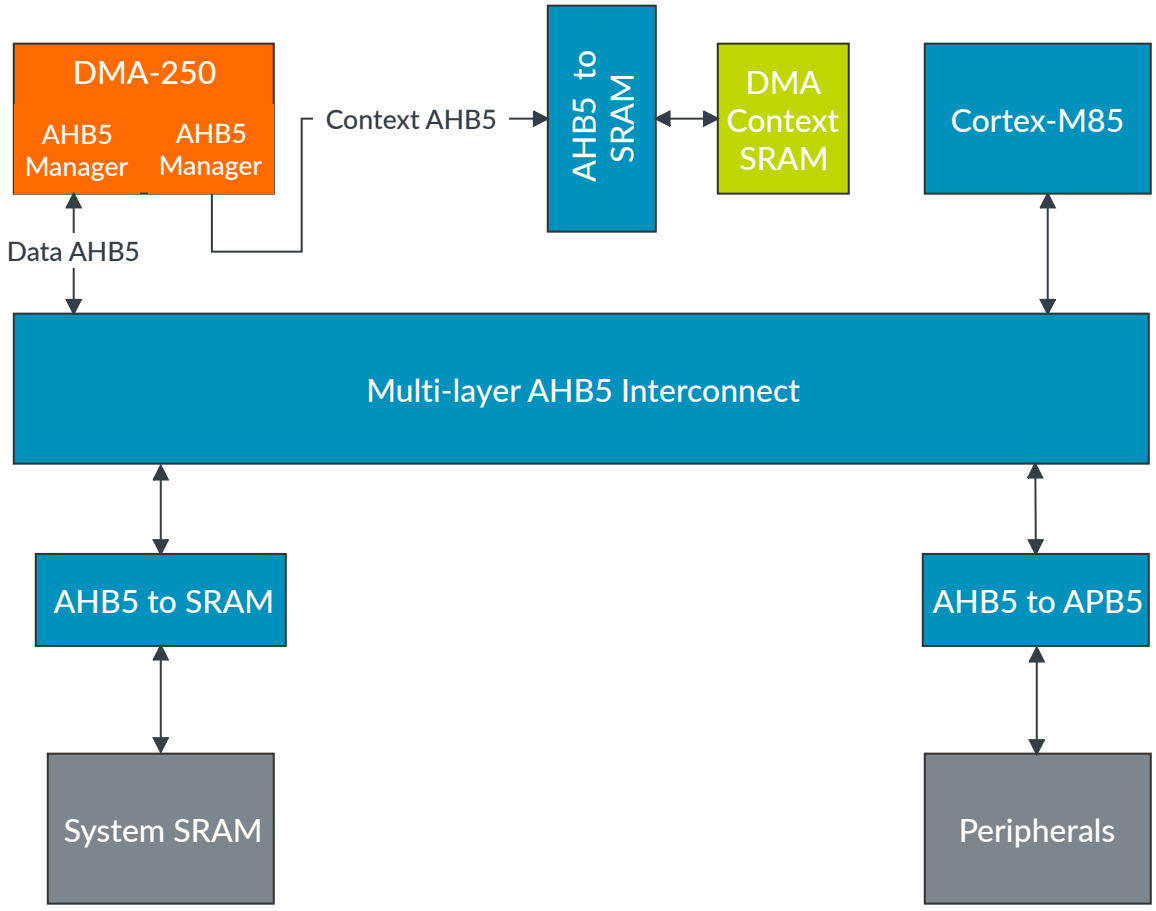
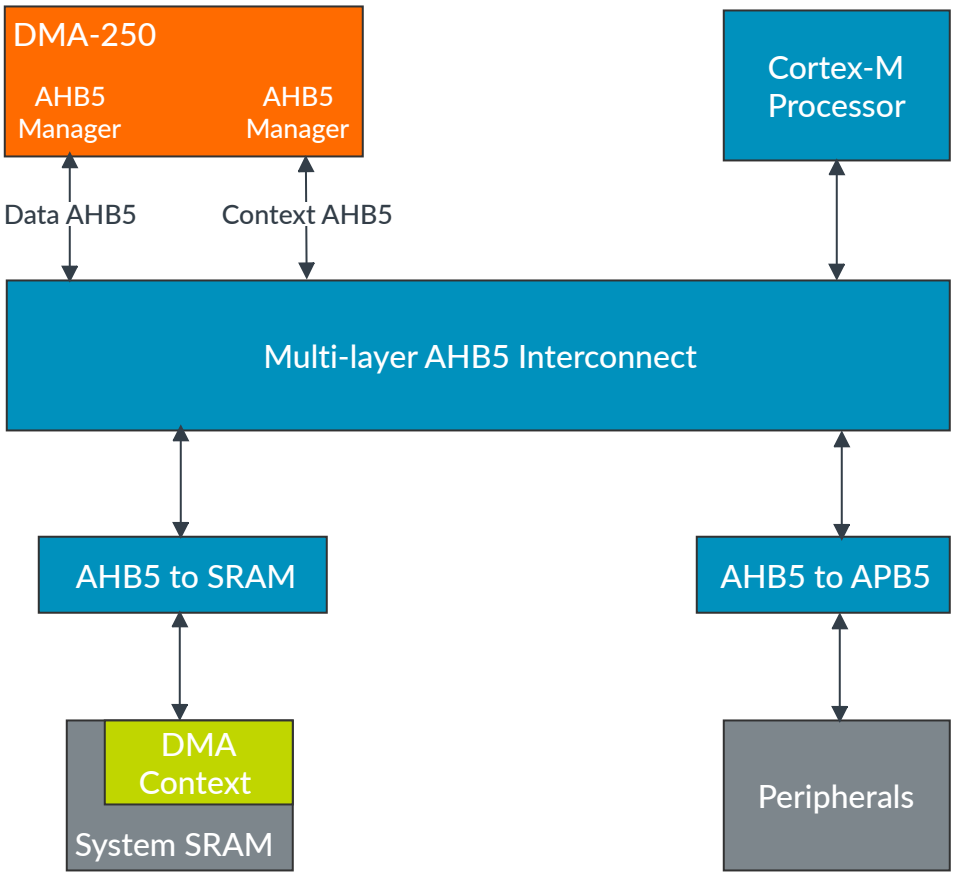


Figure 3-2: No separate SRAM dedicated to context data



When TrustZone security support is configured, DMA-250 implements the hnonsec* signal on its AHB5 interfaces. TrustZone support can be configured using the SECEXT_PRESENT configuration parameter.

APB write and write accesses that target registers in the Context memory are bridged from APB5 to Context AHB5. With the packed context memory map, only a portion of a 32-bit Context memory word must be writable to avoid read-modify-write sequences on C-AHB5. Therefore, when forwarding APB5 write accesses, Context-AHB5 utilizes the narrow write access capabilities of AHB5, with hsize_ctxif and haddr_ctxif determining which byte lanes are active.

Warning

Care should be taken when connecting the Context-AHB5 port and APB port of the DMA to the same busmatrix. A deadlock scenario could occur in the following situation:

- The processor accesses a DMACH configuration register of the DMA (For example: CH_XSIZE), which belongs to a deallocated virtual channel.
- To fulfill the APB request the DMA initiates an AHB5 transfer to access the targeted register in the Context memory area. (Until it is finished the DMA stalls the APB transfer by pulling the PREADY low).

- If the busmatrix is created in a way that the outgoing AHB5 transfer requires the same resource that is already in use by APB transfer, then a deadlock scenario occurs.

To prevent this, make sure that Context-AHB5 and APB ports of the DMA are connected to 2 separate ports of the busmatrix, which can operate in parallel.

3.4 Trigger Interfaces

The Trigger Interfaces can be used to control the interaction of the DMA channel operation with other peripherals.

An external peripheral or device can control the execution of DMA commands through the trigger input interface. The DMAC can also control external peripherals through its trigger output interface.

The trigger input and trigger output interfaces are compatible with each other. Because of this compatibility, it is also possible to connect two DMA channels together or by chaining multiple DMA controllers together.

In addition to the hardware Trigger Interfaces, software Trigger Interfaces are provided through channel control registers that enable the software to interact with triggering.

Related information

- Software triggers
- DMAC operation triggers
- Trigger input flow control mode
- Trigger input command mode

3.4.1 Trigger input interface

The trigger input interface can synchronize the operation of a DMA command and a peripheral. The peripheral can signal DMA-250 when a command or part of a command can be started.

This interface is built up from a simple 4-phase handshake with additional qualifier bus signals that define the type of the current request and acknowledge pair. The interface is defined to be easily compatible with existing Trigger Interface types. The 4-phase handshake also allows simple clock domain crossing structures when the interface is used across different asynchronous clock domains.

3.4.2 Trigger input interface signals

When the req signal is pulled HIGH by the peripheral, it signals that the DMAC can start its command or part of its command.

The DMAC acknowledges the receipt of the *req by asserting the *ack HIGH.

There are additional qualifier signals accompanying both the req and ack signals. These signals give additional information to the receiving entity.

Using the *reqtype[1:0] signals, the peripheral can supply more detailed information to DMA-250. The DMAC uses the *acktype[1:0] signals to give the peripheral more information along with the *ack.

The following signals are defined by the interface:

Table 3-3: Trigger request types

Value	Name	Description
0b00	SINGLE	The peripheral can ask for a single beat of the defined transfer size.
0b10	BLOCK	The peripheral can ask for a larger block of transfer size elements. The block size is defined in the DMAC and in the peripheral as well and can comprise multiple bursts.
0b01	LAST SINGLE	When the peripheral is the flow controller of the transfer and the DMAC does not know the size of the transferable data, the peripheral can close the transfer by requesting a final single beat.
0b11	LAST BLOCK	Similar to LAST SINGLE but the final transfer request is the block size defined in the DMAC and the peripheral. This is used when the complete transfer size is divisible by block size operations.

DMA-250 shows the peripheral that the ack signal received the request. The additional qualifier bus signals, acktype, provide the peripheral with extra information that the trigger request was accepted or not, or if the DMAC transfer is finished after the current transaction.

Table 3-4: Trigger acknowledgment types

Value	Name	Description
0b00	OKAY	This acknowledge type indicates the DMAC accepted the request type. The DMAC indicates that it executes the transfer, but not necessarily at the time of the acknowledge. The DMAC may accept multiple requests before executing the requested operation.
0b10	LAST OKAY	This acknowledge type indicates that the DMAC accepted the request type and tells the peripheral that this is the final transfer of the operation. This acknowledge type has two purposes: • The DMAC acknowledges a LAST request from the peripheral to show that both ends are in sync. • The DMAC is the flow controller and tells the peripheral that no more transfers are expected, so the peripheral needs to reset its transfer counters. For example, a shorter burst is sent as a final transfer.
0b01	DENY	This acknowledge type indicates that the DMAC cannot accept this request now. This can be useful when the DMAC is the flow controller and the peripheral requests SINGLE transfers, but the DMAC expects more transfers to accumulate and optimizes for a block transfer. In this case, the DMAC denies the SINGLE requests so that the handshake can go back to IDLE state.
0b11	RESERVED	Not used.

Related information

- Trigger signals

3.4.3 Trigger input acknowledge types

The DMAC acknowledges trigger requests with different acknowledge types on the acktype signal depending on the trigger configuration, the trigger request type and the remaining number of transfers.

The following tables describe the acknowledge types given by the DMAC in different trigger modes.

Table 3-5: Off (trigger inputs disabled)

Req type	Remaining number of transfers (N)	Ack type	Description
N/A	N/A	N/A	If requests arrive without the DMA using the trigger, then the requests are kept pending

Table 3-6: Command trigger

Req type	Remaining number of transfers (N)	Ack type	Description
N/A	N/A	OKAY	Request type does not matter, the command will start when the request is acknowledged. Additional requests are kept pending.

Table 3-7: DMA Flow control

Req type	Remaining number of transfers (N)	Ack type	Description
SINGLE/ LAST SINGLE	N >= TRIGINBLKSIZE > 1	DENY	The DMAC expects a burst request so it denies the single requests from the peripheral. Last indication on the request signal is ignored for DMA flow control mode.
SINGLE/ LAST SINGLE	N > TRIGINBLKSIZE = 1	OKAY	The block size setting limits the DMAC to single transfers, so it accepts the request. Last indication on the request signal is ignored for DMA flow control mode.
SINGLE/ LAST SINGLE	TRIGINBLKSIZE > N > 1	OKAY	The DMAC expects the last few transfers to be executed with single requests as they do not add up to one whole block, so it accepts the request. Last indication on the request signal is ignored for DMA flow control mode.
SINGLE/ LAST SINGLE	N == 1	LAST OKAY	The final single request is responded with a LAST OKAY indication that tells the peripheral that the command is finished. Last indication on the request signal is ignored for DMA flow control mode.
BLOCK/ LAST BLOCK	N >= TRIGINBLKSIZE	OKAY	The DMAC expects a burst request which matches with the peripherals request type and executes the transfer of the whole block. Last indication on the request signal is ignored for DMA flow control mode.
BLOCK/ LAST BLOCK	TRIGINBLKSIZE > N >= 1	LAST OKAY	Even though the DMAC expects single requests, the peripheral may send a final block request for the remaining data items. This is accepted by the DMAC that executes a final transfer that is smaller than the configured block size. The DMAC indicates that this is the final transfer of the command. Last indication on the request signal is ignored for DMA flow control mode. NOTE: When N==1, both SINGLE and BLOCK requests can finish the command.

Table 3-8: Peripheral Flow control

Req type	Remaining number of transfers (N)	Ack type	Description
SINGLE	N > 1	OKAY	The DMAC executes a single transfer for every request. The DMAC still counts the number of transfers, but it is used as a maximum that limits the command size.
SINGLE	N == 1	LAST OKAY	The DMAC executes the last single transfer. It indicates that the maximum is reached by sending a LAST OKAY and expects no more data transfers. This is probably an unwanted overflow error case on the peripheral side.
BLOCK	N >= TRIGINBLKSIZE	OKAY	The DMAC executes a block transfer for every request. The DMA still counts the number of transfers, but it is used as a maximum that limits the command size.
BLOCK	TRIGINBLKSIZE > N >= 1	LAST OKAY	The DMAC executes the last block transfer which is smaller than a normal block size. It indicates that the maximum is reached by sending a LAST OKAY and expects no more data transfers. This is probably an unwanted overflow error case on the peripheral side. When N==1 both SINGLE and BLOCK requests can finish the command.
LAST SINGLE	N >= 1	LAST OKAY	The DMAC executes the last single transfer. It indicates that the command finished the data movement by sending a LAST OKAY.
LAST BLOCK	N >= TRIGINBLKSIZE	LAST OKAY	The DMAC executes the last block transfer. It indicates that the command finished the data movement by sending a LAST OKAY.
LAST BLOCK	TRIGINBLKSIZE > N >= 1	LAST OKAY	The DMAC executes the last block transfer which is smaller than a normal block size. It indicates that it the command finished the data movement by sending a LAST OKAY.

3.4.4 Trigger output interface

The trigger output interface can signal a connected peripheral that DMA-250 finished a command.

This interface is built up from a simple 4-phase handshake. The interface is defined to be easily compatible with existing Trigger Interface types. The 4-phase handshake also allows simple clock domain crossing structures when the interface is used over different asynchronous clock domains.

3.4.5 Trigger output interface signals

DMA-250 signals to the external peripheral that a command reached its final state by pulling req signal HIGH. The peripheral must acknowledge receiving this information by pulling the ack signal HIGH. The execution of the DMA command is not completed until this acknowledge is received, which gives a synchronization point between DMA-250 and the peripheral.

Related information

- Trigger signals

3.5 LPI interfaces

The DMAC supports low-power integration through the LPI interfaces for both clock and power.

The Q-Channel interface provides quiescence capability for the clock and the P-Channel interface allows power management when the DMAC is IDLE and has no activity ongoing.

The DMAC indicates through its qactive and pactive outputs when it requires the clock to be active (qactive) and the minimum power state (pactive) necessary for the DMAC. The DMAC either accepts or denies the power and clock controller requests based on its current internal state.

3.5.1 LPI power P-Channel

The power P-Channel interface is used to request power quiescence from the DMAC. A power controller drives the request while the DMAC either accepts or denies the request based on its current internal state. The DMAC can also request power for an activity over the pactive signal.

pactive supports a 10-bit wide activity indication from Off to Warm reset. However, the DMAC only asserts Off, Full retention mode and On indications.

The DMAC is not designed to support forceful power shutdown. When a request occurs and activity is ongoing, the DMAC simply denies the request and continues its operation. The request has no effect on the operation except the P-Channel handshake.

Related information

- Power P-Channel
- LPI P-Channel signals

3.5.2 LPI clock Q-Channel

The clock Q-Channel interface is used to request clock quiescence from the DMAC. A clock controller drives the request while the DMAC either accepts or denies the request based on its current internal state. The DMAC can also request clock for an activity over the qactive signal.

The DMAC is not designed to support forceful clock shutdown. When a request occurs and activity is ongoing, the DMAC simply denies the request and continues its operation. The request has no effect on the operation except the Q-Channel handshake.

Related information

- LPI Q-Channel signals

3.6 Control and status interface

The control and status interface makes system control of the DMAC operation (stop and pause) possible and provides status information of the DMAC operation.

3.6.1 General purpose outputs

The *General Purpose Output* (GPO) ports provide extra bits that you can set to a stable value throughout a DMAC operation.

You can use the GPO ports for multiple purposes:

- Selecting memory banks
- Indicating a channel operation on a debug pin
- Controlling external hardware entities, and more.

GPOs are driven per channel so each channel has its own ports.

The value of the GPO is software controlled. The value on the output becomes active when the ENABLE register bit is set and remains stable while the DMA channel is active. The GPO stays on the last value set by the command when the channel stops driving the GPO.


Note

The GPO can be set back to the initial value by running an empty command that selects all GPOs and clears them to 0.

Related information

- CH_GPOEN0

3.6.2 Stop and pause control

The allch_stop* signals can be used to stop the operation of all the active channels of the DMAC by an external hardware unit. The stop function can be useful when dealing with error scenarios in the system and immediate action must clear the DMAC tasks.

The stop waits for all the outstanding responses from read and write transactions and it tries to finish the channel operation as soon as possible by not sending more requests out and not asserting triggers.

These operations are built up from a simple 4-phase handshake. When the allch_stop_req_nonsec is asserted, all Non-secure channels that are not in IDLE are stopped. The software can enable channels but they are immediately stopped when this request is asserted. The assertion of allch_stop_ack_nonsec signals that all Non-secure channels have been stopped or are inactive. When the security option is enabled (SECEXT_PRESENT=1) separate signals, allch_stop_req_sec

and allch_stop_ack_sec, exist for the Secure channels that operate on the channels in the Secure domain only.

The operation of the DMA channels can be paused immediately at a point in time by the allch_pause signals operated by an external hardware unit. The pause function can be useful to freeze the channel in a state and check the current values of its registers, or to pause the DMA unit for a while to free up bus infrastructure resources. The enable bit remains asserted and every transfer and trigger state are kept, no information is lost. These operations are built up from a simple 4-phase handshake. When the allch_pause_req_nonsec asserted, all Non-secure channels that are not in IDLE are paused.

The software can enable channels but they are immediately paused when this request is asserted. When the request is deasserted, the operation continues. When the allch_pause_ack_nonsec asserted, all Non-secure channels are paused or inactive. When the security option is enabled (SECEXT_PRESENT=1) separate signals, allch_pause_req_sec and allch_pause_ack_sec, exist for the Secure channels that operate on the channels in the Secure domain only.

Related information

- Status and Control signals

3.6.3 Cross Trigger Interface

The DMAC provides a Cross Trigger Interface (CTI) that allows pausing and resuming all channels at once. The CTI is required in a system where a processor is halted for debug purposes and the debugger must save the actual memory contents so the DMAC can also be paused to avoid corrupting the current state of the system.

Related information

- Table A-11: Cross Trigger Interface signals on page 170

3.6.4 Status signals

DMA-250 provides hardware signals on its top-level that indicate the actual status of the channels. These signals can be used by system control or other external logic.

Each channel has various 1-bit status indicator signals assigned to them that reflect if the channel is currently enabled, stopped, paused or encountered error during operation. Tthe signals also show the security and privilege status of the channel. These signals are aggregated to vectors the width of NUM_VCH, where the nth bit shows channel #n's corresponding status.

When SECEXT_PRESENT is set to 1 the ch_enabled, ch_err, ch_stopped, ch_paused and ch_priv signals must be separated, based on the information in the ch_nonsec signals, into two signal groups to keep the secure and non-secure status information in their correct security domain.

The ch_nonsec signals are not present when SECEXT_PRESENT is set to 0.

Related information

- Table A-10: Status signals on page 170

3.7 Configuration interface

The behavior of the DMAC can be configured through static input ports. Static input ports come from either tie-off signals or are driven by configurable registers that are stable and contain a valid value when the DMAC is released from reset.

For more details, see Configuration signals.

Automatic boot interface

Assertion of the boot_en indicates that automatic booting is enabled and the other signals are valid. If boot_en is not asserted, automatic booting is disabled and the other boot signals are ignored. All boot signals must be stable when deasserting the reset and remain stable until fetching of the boot command is started.

For more detail on automatic booting, see Automatic boot feature.

Related information

- Configuration

4. DMAC operation

The DMAC can be configured by register writes to execute a vast variety of commands.

When the software finalizes the register settings and the command link elements, the channel can be enabled by an additional register write. This step makes all configuration registers become read-only from the software side. The DMAC then starts checking the validity of the command and executes the transfers based on the register settings.

The commands can contain simple memory transfers, scatter-gather type transfers, handling of triggers, general purpose outputs, and many other combinations. When a command is finished the DMAC either returns to idle, reloads the command or jumps to a next item in a command list. When the DMAC returns to an IDLE state it can generate interrupts and trigger output signals to synchronize this event with software or other hardware elements in the system. When the DMAC is in IDLE, the registers can be configured again.

The DMAC is also prepared to detect errors during the memory transfer. When a fault is received, it returns an error and stops its operation.

4.1 DMAC operation basic commands

This section contains an overview of DMA-250 operation basic commands.

Transfer type 1D

Transfer type 1D is a one-dimensional block transfer. It transfers data from the source address to the destination address continuously in configured transfer size. This mode is the typical use-case for bulk data transfer where no data formatting is necessary during DMAC transfer. Transfer size setting ranges from byte to bus width in power of 2 increments.

The transfer size is the same on both source and destination sides. The source and destination addresses are aligned with the transfer size as the lower address bits are ignored according to the TRANSIZE value. The number of transfers to copy are specified in transfer size increments. The software can also limit the maximum burst length the DMAC can send to the bus to allow arbitrating others on the interconnect.

Figure 4-1: 1D transfer example

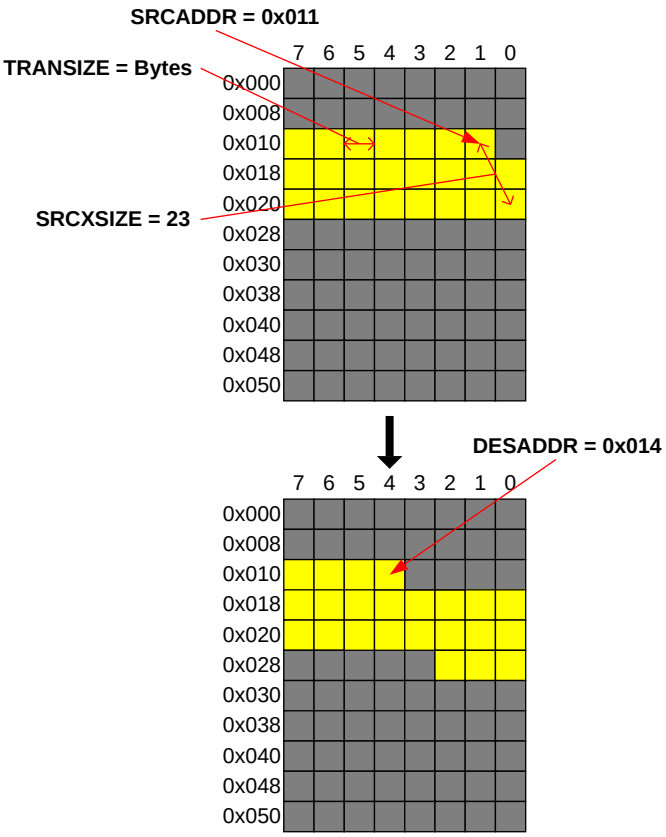


Table 4-1: 1D transfer attributes

Parameter name	Register map entry	Description
Source Address	CH_SRCADDR	The starting source address to transfer data from. The source address must be aligned with the transfer size.
Destination Address	CH_DESADDR	The starting destination address to transfer data to. The destination address must be aligned with the transfer size.
Source X-size	CH_XSIZE.SRCXSIZE	The source number of transfers in the X dimension. The width of the source transfers is equal to the TRANSIZE value of the source transfers.
Destination X-size	CH_XSIZE.DESXSIZE	The destination number of transfers in the X dimension. The width of the destination transfers is equal to the TRANSIZE value of the destination transfers.
Transfer size	CH_CTRL.TRANSIZE	The data width that is utilized by both the source and the destination of the DMAC operation.
Priority	CH_CTRL.PRIORITY	This signal is used for arbitration between the channels.
Source maximum burst size	CH_SRCTRANSCFG.SRCMAXBURSTLEN	The maximum burst length that is supported on the source side of the DMAC operation.
Destination maximum burst size	CH_DESTRANSCFG.DESMAXBURSTLEN	The maximum burst length that is supported on the destination side of the DMAC operation.

Parameter name	Register map entry	Description
Source AHB5 transfer properties	CH_SRCTRANSCFG.SRCTRP	Source transfer properties These properties include memory type, shareability attribute, secure attribute and privilege attribute.
Destination AHB5 transfer properties	CH_DESTRANSCFG.DESTRP	Destination transfer properties These properties include memory type, shareability attribute, secure attribute and privilege attribute.
1D wrap type	CH_CTRL.XTYPE	Determines whether data transfers from source to destination should take place based on address, address increment and size setting, or if the command is an empty command with no data transfers.
Source address increment	CH_XADDRINC.SRCXADDRINC	Increment used to calculate address on source side. The increment is: *TRANSIZE * SRCXADDRINC*
Destination address increment	CH_XADDRINC.DESXADDRINC	Increment used to calculate address on destination side. The increment is: *TRANSIZE * DESXADDRINC*

Related information

- DMACH<n> summary, DMA Channel Register Frame

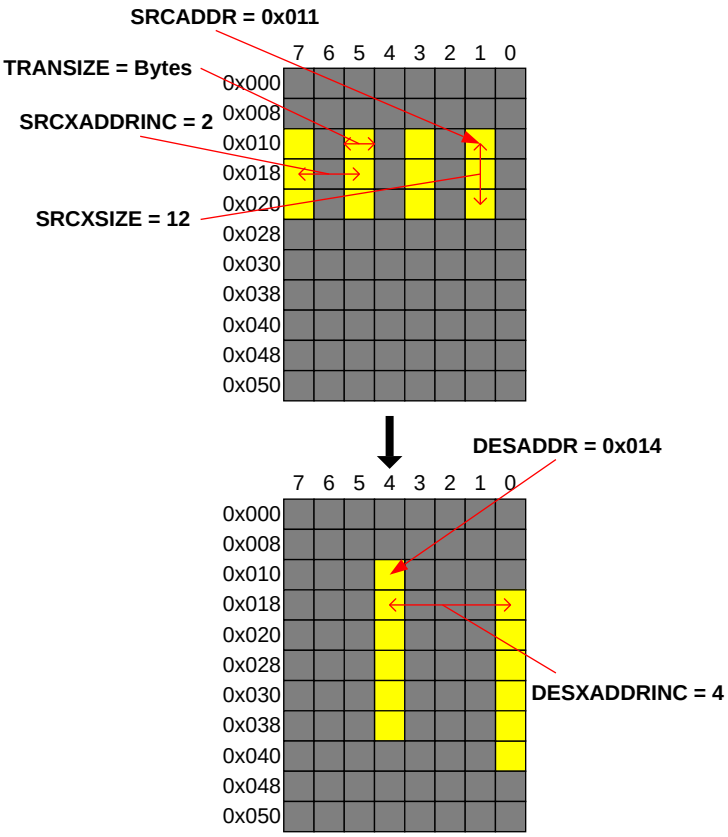
4.1.1 1D transfers with increments

1D transfers also have increment capabilities to the source and destination addresses. Non-adjacent addresses are also supported, the increment value can be configured separately for source and for destination addresses.

1D transfers also allows setting 0 increment to result in a peripheral like access targeting a single memory location with multiple accesses.

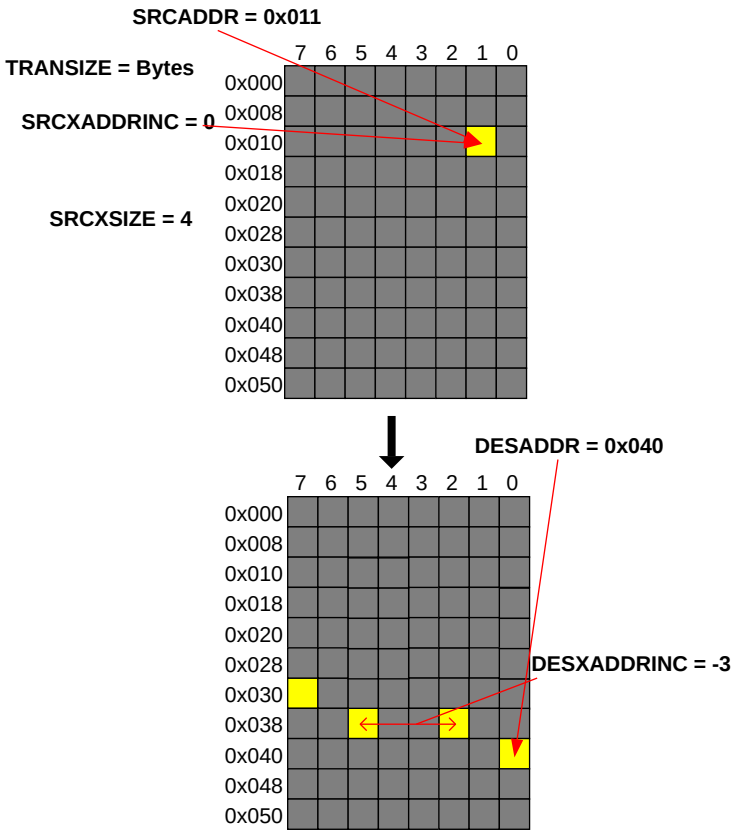
The number of transfers is the same on both sides. Increments are based on the *transfer size*.

Figure 4-2: 1D transfer with increments



The increment value can be negative, too, in case a command must fetch or fill the memory in a different direction. The following example shows a transfer of 4 bytes from the same location to the destination where reverse addressing is used with decrements of 3. All bytes are read from the address `0x11`, the first byte is stored at `0x40`, the second at `0x3D`, the third at `0x3A`, and the final fourth at `0x37`.

Figure 4-3: 1D transfer with zero and negative increments



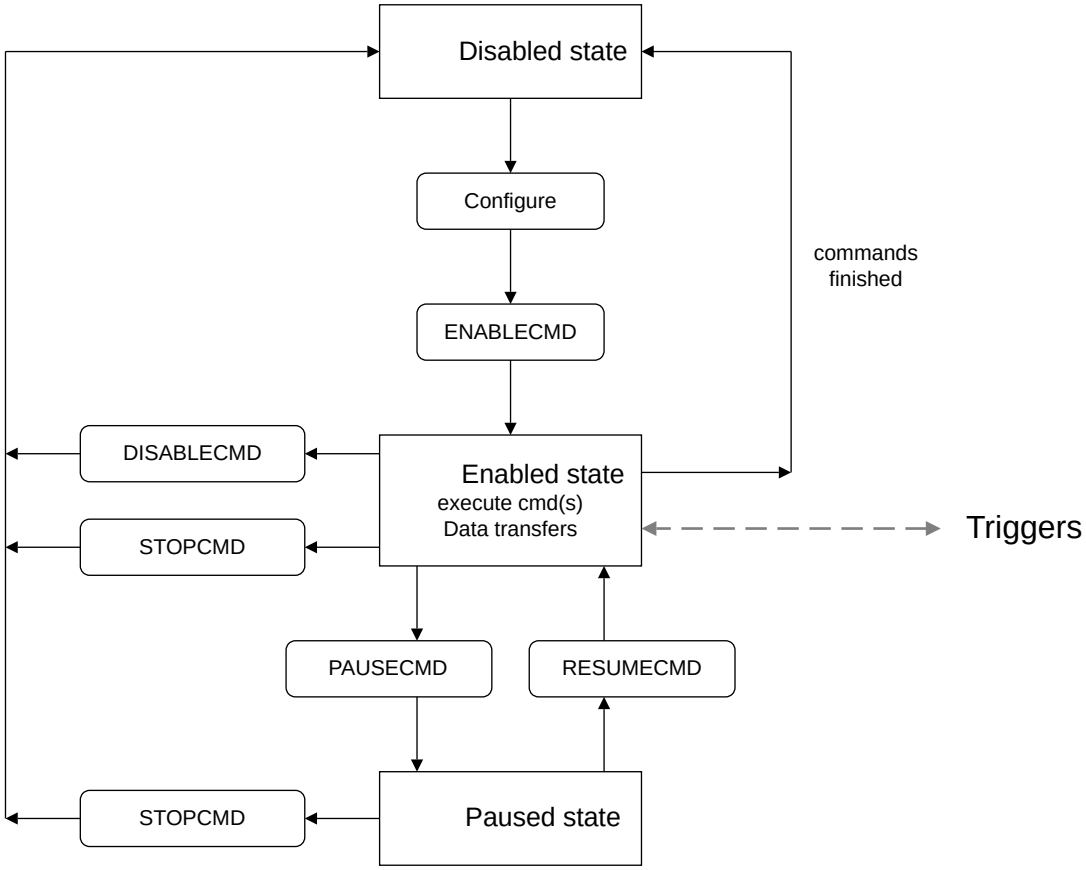
DMA-250 supports 16-bit wide increment registers. The range is interpreted a 2s complement where negative numbers range from -32768 (`0x8000`) to -1 (`0xFFFF`) and zero or positive numbers range from 0 (`0x0000`) to 32767 (`0x7FFF`).

4.2 DMA Channel lifecycle

THE DMA channel has threee possible states: Disabled, Enabled, and Paused.

The operation of a DMA channel is shown in the following figure:

Figure 4-4: DMA channel lifecycle



Disabled state

After reset, the channel is in the disabled state. The command to be executed must be set up by configuring the channel registers. The software can modify every writable configuration register in this state.

After the channel is configured, the execution is started by setting the ENABLECMD bit in the CH_CMD register.

Enabled state

Execution of the command happens in the Enabled state. When the ENABLECMD bit is set, the ch_enabled signal is asserted to indicate ongoing command execution. This status signal remains HIGH until the command is completed and no other command linked command is configured, or the channel operation is stopped manually or because of an error event.

In the enabled state, all configuration registers of the channel become hardware-controlled ones. The software is still allowed to read the registers, but the hardware is updating their contents throughout the operation of the command. The software can adjust during the command operation

only the CH_CMD and CH_STATUS registers that allow the software to stop or pause the command execution, clear interrupts and inspect internal work register states.

The *ADDR registers point to the next location in the memory where accesses are made. The *SIZE registers count down from the starting value to 0 and show the remaining number of transfers when read.

The command execution ends in the following cases:

- The command is finished successfully.
- An error event (for example, configuration error) causes the command to stop.
- The channel is disabled because of setting the DISABLECMD bit in the CH_CMD register.
- The channel is stopped because of:
  - Setting the STOPCMD bit in the CH_CMD.STOPCMD register
  - Setting the ALLCHSTOP bit in the NSEC_CTRL or SEC_CTRL register (depending on the configured security of the channel)
  - Asserting the allch_stop_req_nonsec or allch_stop_req_sec signal (depending on the configured security of the channel)

The channel returns to the disabled state when any of the previous conditions cause the command to end its execution. The status flags (STAT_* fields in the CH_STATUS register) are set when returning to disabled state. The software can inspect the status flags to determine whether the execution was successful or an error occurred.

Paused state

The channel enters the paused state when a pause request is issued. A pause request can be issued by:

- Setting the CH_CMD.PAUSECMD bit (software pause request).
- Setting the ALLCHPAUSE bit in the NSEC_STATUS or SEC_STATUS register (depending on the configured security of the channel).
- Asserting the allch_pause_req_nonsec or allch_pause_req_sec signal (depending on the configured security of the channel).
- Automatic software pause request when the STAT_DONE flag is asserted and the channel has been preconfigured to enable done-pause by setting the CH_CTRL.DONEPAUSEEN bit.
- Cross Trigger Interface
- Warm reset

In the paused state, the channel execution is stalled. The bus transactions are stopped and processing of triggers is also halted. The software can inspect the channel state while being paused, as the address and size registers reflect the current state of the channel.

Operation of the channel is continued when the pause request is revoked:

- Software pause request, including done-pause, can be revoked by setting the CH_CMD.RESUMECMD bit.

- All-channel-pause request is revoked by clearing the STAT_ALLCHPAUSED flag in the NSEC_STATUS or SEC_STATUS register.
- Hardware pause request is revoked by deasserting the allch_pause_req_nonsec or allch_pause_req_sec signal.

The channel can be stopped in the paused state by issuing a stop command.


Note

The channel remains enabled the (ch_enabled signal is asserted) in the paused state, only the execution is halted temporarily. The configuration registers remain read-only when paused.

Related information

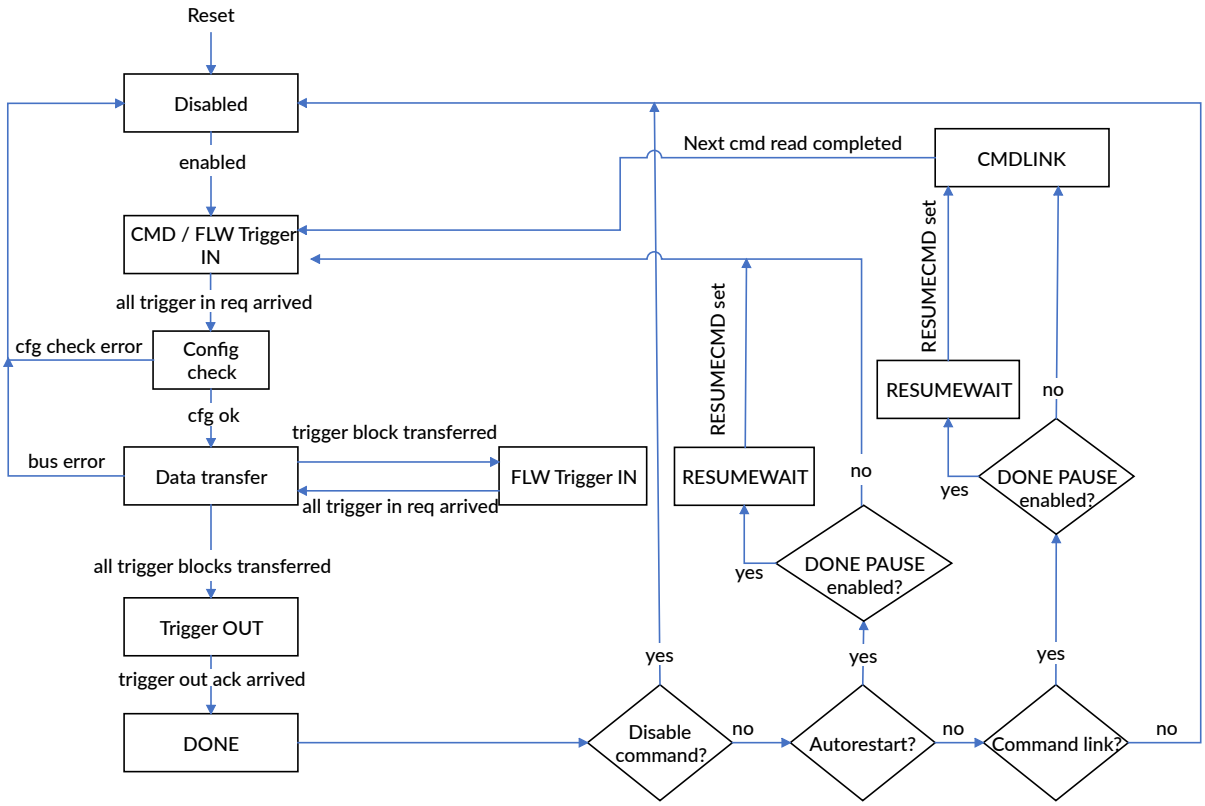
- Status and Control signals

4.2.1 Command execution states

Each command has various steps that are executed in a fixed order. Some steps can be skipped depending on the configuration of the command.

The following flowchart illustrates the DMA channel command execution states.

Figure 4-5: DMA channel command states



Disabled state

This is the default state. The channel is not active, no command execution is taking place. Software can set up a command and channel properties by programming the channel registers. When setup is completed, command execution can be started by enabling the channel. The GPO output is set to the new value (if configured) before proceeding to the next state.

Command/Flow trigger input state (CMD/FLW trigger IN)

This state handles the incoming trigger requests when trigger inputs are configured for the command. The channel execution is halted until all expected triggers request (regardless of command or flow mode) are received on the source and/or destination trigger inputs. If no trigger inputs are configured, this step is ignored.

Configuration checking state (Config check)

This state verifies the configuration of the channel before starting data transfer. If an invalid register value or conflicting settings are detected, the corresponding error status flags are set, and the channel operation is stopped.

Data transfer state

The data transfers occur in this state. Flow-control trigger are processed only in this state.

Flow trigger state (FLW Trigger IN)

After a transferring the programmed sized block and li flow triggers are configured, command execution is halted until all expected flow trigger requests (source and / or destination) are received.

Trigger output state (Trigger OUT)

As the last step of a command, the trigger output request is asserted, if output triggering is configured. Operation halts until an acknowledgment is received. If no output triggering is configured, this step is skipped.

Done state

Reaching this state indicates that the command has completed successfully. The CH_STATUS.STAT_DONE flag is set depending on the configured CH_STATUS.DONE_TYPE field.

If a CH_CMD.DISABLECMD has been issued up to this point, the channel execution is finished and the CH_STATUS.STAT_DISABLED flag is set.

If CH_AUTOCFG.CMDRESTARTCNTis nonzero or CH_AUTOCFG.CMDRESTARTINFEN is set, then the current command is restarted automatically. The CH_AUTOCFG.CMDRESTARTCNT counter is decremented by 1 if it is nonzero.

If no automatic restart of the current command is configured, then the CH_LINKADDR.LINKADDREN flag is checked. If set, the command link feature is enabled, and the next command is fetched using the LINKADDR pointer.

When a trigger input request is sent to a DMA channel after the channel has received the last trigger it was expecting, the trigger request remains pending. The next linked command that is configured to use the trigger input can respond to it, and therefore no trigger event is lost.

If the auto restart and command link are both disabled, the channel operation is finished, and the channel returns to disabled state.

Otherwise, the CH_CTRL.DONEPAUSEEN flag is checked. If set, the channel execution is paused until a RESUMECMD is received.

Related information

- CH_CMD
- CH_AUTOCFG
- CH_LINKADDR

4.2.2 Automatic restart of commands

The auto restart feature improves the efficiency when a single command must be repeated many times in a loop, by eliminating unnecessary reloading of the same command.

Auto restart can be enabled by setting the CH_AUTOCFG.CMDRESTARTINFEN or CH_AUTOCFG.CMDRESTARTCNT fields to a nonzero value.

The number of iterations depends on the settings of these fields:

- CMDRESTARTINFEN is set: The automatic restarting keeps occurring in an infinite loop until the channel is disabled or stopped.
- CMDRESTARTINFEN is not set: The automatic restarting occurs for the number of times that is configured in CMDRESTARTCNT field. If it is configured to zero, the auto restart feature is not enabled.

If the auto restart feature is enabled, the destination and source addresses and the transfer size settings are reloaded with their original value depending on the setting of the REGRELOADTYPE field. If an address is not reloaded, the starting address of the next iteration of command execution is defined based on the operation type.

Related information

- CH_AUTOCFG

4.2.3 Error handling

There are three types of errors that can occur during DMAC operation:

- AHB5 bus error during operation
- Configuration error
- Context AHB5 errors

Any of these errors cause the DMA channel to stop its operation. The error status flag, STAT_ERR, is set in the CH_STATUS register when the channel state transitions to DISABLED (ch_enabled = `0b0`).

If the related interrupt enable flag (INTREN_ERR) is set, the interrupt flag (INT_ERR) is also asserted.

The cause of the error is indicated in the CH_ERRINFO register which contains valid information when STAT_ERR is asserted.

The following table lists the possible errors indicated through the CH_ERRINFO register.

Table 4-2: DMA-250 channel errors

Error type	CH_ERRINFO field	bit offset	Description
Bus error	BUSERR	[0]	Error received on the Data AHB5 interface.
Configuration error	CFGERR	[1]	The command that is about to be executed has an invalid configuration in the registers.
Context Memory access error	CXTERR	[8]	Error received on the Context AHB5 interface. See Context AHB5 errors
Additional error information	ERRINFO.LINKHDRERR	[24]	Configuration error because an invalid command-link header. Indicates that an empty (all bits zero) command-link header was read.
	ERRINFO.REGVALERR	[25]	Configuration error because one of the configuration registers is set to an illegal value. See Table 4-3: DMA-250 channel error reasons - ERRINFO fields on page 50.
	ERRINFO.CFGCONFLERR	[26]	Configuration error because there are conflicting settings in the configuration registers.

The following ERRINFO fields give additional information on the reasons for the errors indicated in the CH_ERRINFO register. Bit offsets are relative to CH_ERRINFO register.

Table 4-3: DMA-250 channel error reasons - ERRINFO fields

Bits	Name	Description
[31:27]	-	Reserved, **RAZ/WI**
[26]	CFGCONFLERR	The trigger-in configuration conflicts with other command settings.
[25]	REGVALERR	CH_CTRL.TRANSIZE setting over the DATA_WIDTH limit. The DMA does not support splitting larger elements.
[24]	LINKHDRERR	Configuration error because an invalid command-link header was read.

Data AHB5 bus errors

When a bus error on the Data AHB5 interface is detected, the current DMA command stops immediately and the data associated with the bus error is discarded. All outstanding transactions are completed, but their data is also discarded. The BUSERR field is also set in the CH_ERRINFO register to indicate the source of the bus error condition.

Context AHB5 errors

When an error response on the Context AHB5 interface is detected, the DMA channel that the Context Memory area with the bus error belongs to is stopped immediately. The CTXERR field is set in the CH_ERRINFO register.

As all channels that are in the same security world could be affected by the error, Context AHB5 errors are also reported on DMA unit level in the NSEC_STATUS.STAT_CTXERR or SEC_STATUS.STAT_CTXERR field in the DMA Unit Non-Secure or Secure control Register frame. The specific register depends on the security of the DMA Channel that received the error response.

In case of a Context AHB5 error, the configuration of the affected DMA channel is considered corrupted. Enabling a DMA command on the channel without reprogramming may result in **UNDEFINED** behavior.

Configuration errors

The physical channel (PCH) checks for configuration errors before data transfer execution. No bus transfers are initiated when a configuration error is detected.

The three possible sources of configuration errors are:

- Invalid command link header (LINKHDRERR) - all bits are Read-As-Zero.
- Illegal field value set in a configuration register (REGVALERR).
- Incompatible settings are present in the configuration registers (CFGCONFLERR).

Illegal field value errors

The causes of illegal field value error (REGVALERR) are:

- TRANSIZE field is greater than bus data width.
- SRCXSIZE and DESXSIZE are not equal.

This issue can occur when a channel is stopped mid-operation when the working values of SRCXSIZE and DESXSIZE temporarily deviate, then the channel is re-enabled without updating the XSIZE registers.

Incompatible setting errors

The reasons for conflicting configuration settings (CFGCONFLERR) are:

- The trigger-in configuration conflicts with the configured transfer mode (XTYPE).

See Configuration for detailed conditions on incompatible settings.

4.2.4 Command execution status reporting

The software can read the contents of XSIZE, and the address registers after the DMA command execution has concluded.

The contents of these registers depend on the manner of DMA command stop.

DONE status reached - command execution was completed :

- Size registers are 0, no more transactions to be completed on either read or write side.
- Address registers contain the next address as if there were subsequent DMAC operations of the same type as the completed command.

STOPPED/PAUSED status - command execution was stopped or paused before execution completed :

- Size registers contain the actual coordinate of the current transaction that was not completed.
- Address registers contain the address of the next transaction that was not completed.

In STOPPED/PAUSED state, address and size register values are only approximations of the actual command execution status, and might not contain the exact position. They serve as a hint about where approximately the command execution was halted.

If pausing occurs during command link execution, the registers read back might contain an incoherent command, because not all registers might be updated when the DMA enters PAUSED state.

Address register at the end of a command

At the end of command, address registers contain the address of the next DMAC transaction of the same type that was not completed. If the command is completed, the next address points to the address that would have been used for the transaction if the command contained more transactions.

Addresses after empty commands

If there is an empty command execution, read and write addresses remain unchanged at the end of the DMA command. If either side of the transaction is involved in command execution while the other side remains idle, address registers on the idle side do not change while the address registers on the active side do change according to this behavior.

For example, when write is active but read is inactive in a DMA command, the read address does not change but the write address does change according to the behavior described above.

Size registers at the end of a command

Size registers contain the coordinates of last unprocessed transaction within the current line. Both size registers are 0 when the DMA command completes.

When a stop or bus error interrupts the DMA command, the XSIZE registers contain the remaining unsent DMAC transactions in TRANSIZE.

Size after empty commands

If there is an empty command execution, sizes do not change at the end of DMA command. If either side is involved in command execution while the other remains idle, size registers on the idle side become zero when the command is completed.

4.3 DMAC operation triggers

Trigger inputs and outputs can synchronize activities within a system without SW intervention. The DMAC can have trigger inputs and outputs on a channel basis when enabled. The input triggers support multiple modes for different purposes, while the output triggers be used to mark the final step of a complete DMAC operation.

4.3.1 Trigger inputs

Trigger input ports use the trigger signals to enable peripherals to communicate with the DMAC and sequence DMAC operations without software intervention. Triggers provide flow control for transfers within a DMAC operation and can also be used to start entire DMAC operations. The mode in which the trigger input is used can be selected through configurable registers.

A 4-phase handshake bus is used between the peripheral and DMA-250. Extra qualifier signals extend the req and ack signal pair in both directions. The requesting peripheral can signal to the DMAC that it has data to be transferred by the req signal. The additional qualifier signals, reqtype, are used to show the DMAC that it has single or block data to be transferred, and that it has the last data amount to be transferred.

The different request types can be seen in Trigger input interface signals.

A virtual channel has two trigger input ports when enabled:

- One trigger input port for the source entity.
- One trigger input port to the destination entity.

The number of trigger inputs is fixed based on whether triggers are enabled or not (depends on if HAS_TRIG = 1). If enabled, each channel has 2x input trigger ports: 1x source trigger input and 1x destination trigger input port. If disabled, there are no trigger input ports on the DMA.

Trigger input modes are set through software programmable registers. Trigger related registers can only be adjusted when the DMA channel is idle.

When a DMA command is configured so that the source or the destination side uses triggers, the DMAC will wait for the corresponding trigger request to arrive before starting the DMA transfers. When both source and destination triggers are used, the DMAC will wait for both trigger requests to arrive before initiating any DMA transfers.

If a request arrives on a trigger input port after the DMA channel has already received all expected trigger requests for the current command, or if the current command is not using that particular trigger input, the request signal remains pending without getting any acknowledge. When a channel is configured to use a trigger input port with pending request, the channel receives the req immediately after channel enable.

When a channel is paused, the pending triggers are left as is, and the service is pending. The interface can stall in any state. During a pause, external signals, like a req signal in the trigger input or an ack signal in the trigger output, can change state. After the channel operation is resumed, the trigger operation continues.

The software can initiate a trigger request automatic clear. The DMAC can send a “deny” acknowledge to a pending request on a trigger input. This way, the unneeded pending request can be cleared without trigger protocol violation.

Note

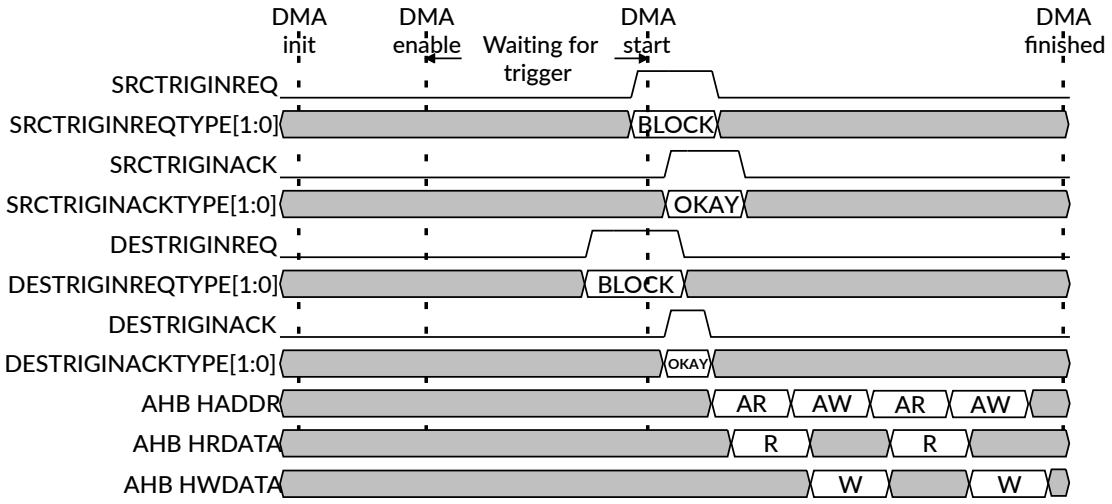
For debug purposes, the software can mimic the external hardware trigger. See Software triggers for more information.

4.3.1.1 Trigger input command mode

A command mode trigger input can be used to initiate full DMA commands. When this mode is used, the request on the trigger input interface is expected only once per command, or once per restart cycle if the command is configured to restart automatically. Once the trigger request arrives and the DMA channel proceeds with the command, the request is acknowledged.

The following figure shows a scenario where both the source and destination sides use command mode triggers. After the DMA command is set up and the DMA channel is enabled, the DMA waits for the trigger requests to arrive.

Figure 4-6: Command trigger for both source and destination sides



The figure shows that the destination trigger is received first. It is not acknowledged until the DMAC detects the source trigger request. At this point, both source and destination triggers are acknowledged and the DMA transfers can start. The DMAC does not accept new trigger requests until the complete DMA operation lasts.

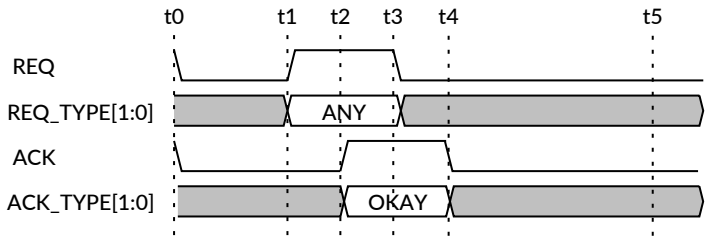
The trigger handshake can return to IDLE at different points in time depending on the driver of the request signal, but it does not interfere with the execution of the data transfers.

The command can fully complete, but if the trigger handshake is still not finished by the time a new command is set up and started, the new command will be stalled before starting any DMA transfers until the trigger handshake is finished. This applies even if the new command does not use triggers.

For command mode triggers, the reqtype[1:0] can take any of the following values: SINGLE/BLOCK/LAST SINGLE/LAST BLOCK.

The acktype[1:0] value of OKAY/LAST OKAY are both valid for command mode triggers. DMA-250 returns OKAY value. The DMA channel acktyp[1:0] value of DENY is not used for command triggering.

Figure 4-7: Trigger input ACK assertion and deassertion for command triggers



The figure shows the ack deassertion approach of the DMAC when using command triggers:

t0

The Trigger Interface is in IDLE.

t1

The peripheral requests any type of trigger to start the command.

t2

The DMAC accepts the request if the DMA channel is waiting for the trigger and ready to execute the operation.

t3

The peripheral deasserts the req signal when it receives the OKAY response from the DMAC. The DMAC still executes the requested operation.

t4

The DMAC detects that the req is deasserted and it deasserts the ack so the Trigger Interface returns to IDLE. The command is still running.

t5

The command is finished but it is not reflected on the trigger input interface. Trigger out signals can be used to show the end of a command.

Related information

- Trigger Interfaces

4.3.1.2 Trigger input flow control mode

Trigger inputs can also be used for flow control purposes and only enable small parts of a larger command for every trigger.

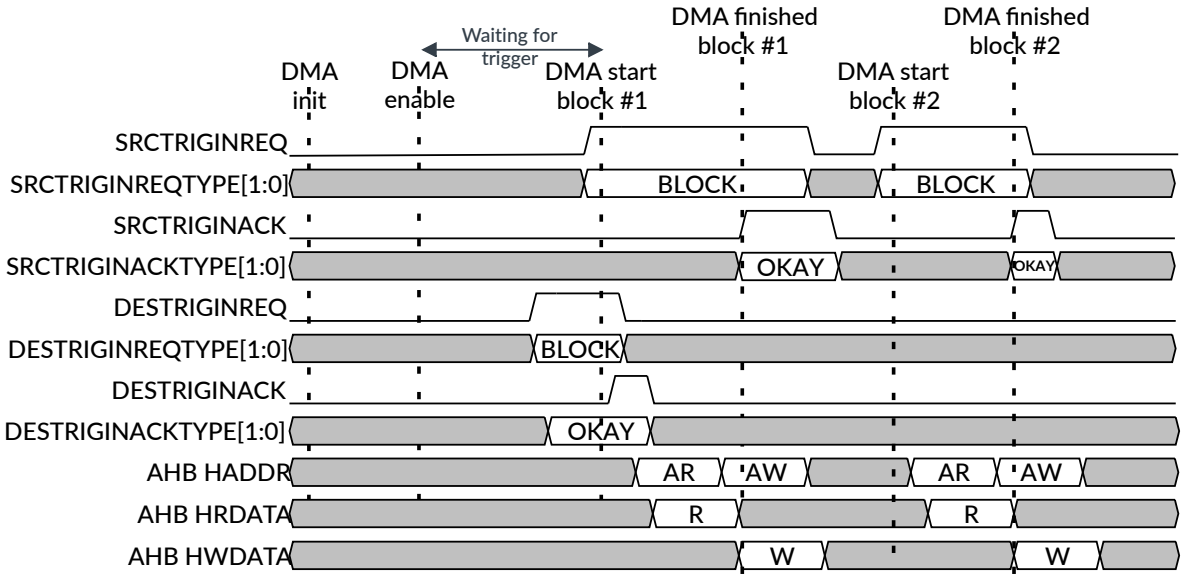
For this mode, the software must set the source and destination trigger size in CH_SRCTRIGINCFG.SRCTRIGINBLKSIZE and CH_DESTRIGINCFG.DESTRIGINBLKSIZE fields respectively, which tells the number of transfer size blocks to be accessed for one trigger event.

The source trigger input is used to control the read flow, while the destination trigger input controls the write flow of a DMA operation. When this mode is used, the request on the trigger input interface is expected for each block to be transferred and is acknowledged after all transfers are completed within the current block, meaning all read transfers in case of source trigger and all write transfers in case of destination trigger.

The following figure shows a scenario where the source trigger is used for flow control and the destination trigger is used as command trigger.

The figure shows that the destination trigger is received first. It is not acknowledged until the DMAC detects the source trigger request. At this point, the DMA transfers can start, but only the destination trigger is acknowledged as a command mode trigger, and the DMA does not accept new destination trigger requests until the complete DMA operation lasts.

Figure 4-8: Flow control mode trigger for source and command for destination

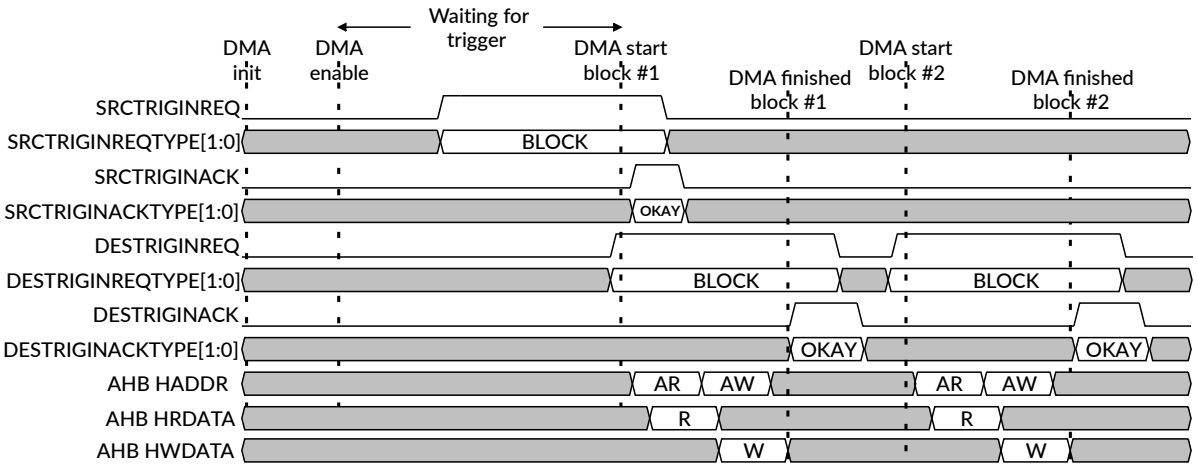


After the DMA command is set up and the DMA channel is enabled, the DMAC waits for the trigger requests to arrive.

The source trigger request is acknowledged after all read transfers are finished for the current block. The DMAC then waits for the subsequent block's source trigger request to arrive. When the DMAC detects the new request, it initiates the DMA transfers for this block. Once all read transfers are finished, the DMAC acknowledges the source trigger request.

The following figure shows a scenario where the source trigger is used as command trigger and the destination trigger is used for flow control.

Figure 4-9: Command trigger for source and block for destination

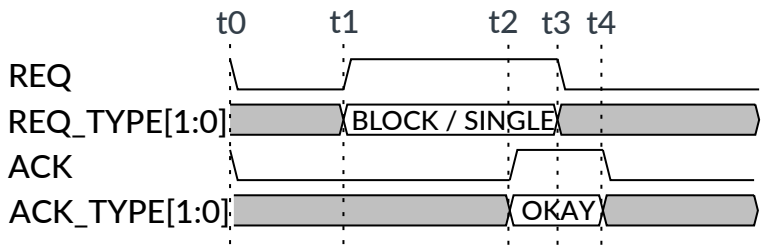


The overall schedule of events is very similar to the previous example. After the DMA command is set up and the DMA channel is enabled, the DMAC waits for the trigger requests to arrive. When the DMAC detects both trigger requests and can proceed with the command, it acknowledges the command trigger, which is now the source side.

The DMAC performs the transfers for the first block and acknowledges the destination trigger request once all write transfers are finished for the current block. The DMAC then waits for the subsequent block's destination trigger request to arrive. When the DMAC detects the new request, it initiates the DMA transfers for this block. Once all write transfers are finished, the DMAC acknowledges the trigger request.

The DMA channel can be set to allow DMAC flow control operation for both source and destination sides. In this mode, the DMA channel counts the number of transfers and, when the required number is reached, it ends the DMA channel operation. The DMA channel sends OKAY acknowledge to the peripheral when there are still transfer size blocks to be transferred. When the last transfer size block is reached the DMA channel sends LAST OKAY acknowledge.

Figure 4-10: Trigger input ACK assertion and deassertion for flow triggers



The figure shows the ack deassertion approach of the DMAC when using flow triggers.

t0

The Trigger Interface is in IDLE.

t1

The peripheral requests a BLOCK or SINGLE trigger type.

t2

The DMAC started the execution of the request as soon as possible and at t2 all transfers requested for this operation are finished.

t3

The peripheral deasserts the req signal when it receives the OKAY response from the DMAC.

t4

The DMAC is finished with servicing the request and deasserts the ack signal when it is ready to accept a new request.

Related information

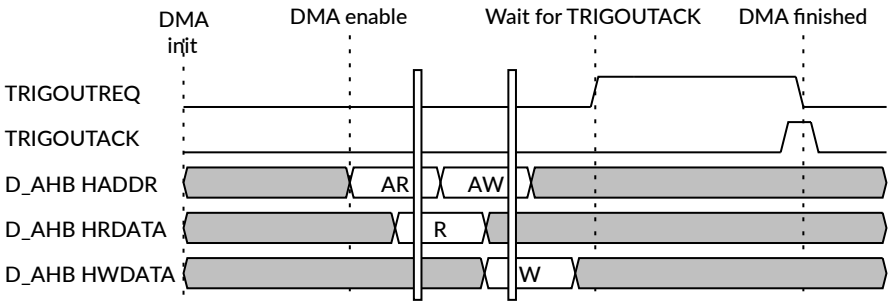
- Trigger Interfaces

4.3.2 Trigger outputs

Trigger outputs define the end of the command execution before the DONE interrupt is asserted. They can be used to synchronize activities between hardware elements in the system.

When waiting for trigger output acknowledge, the channel operation is stalled until the acknowledge is received and trigger output interface of the channel returns back to IDLE.

Figure 4-11: Trigger output request assertion



A channel has one trigger output port when enabled.

Note

For debug purposes, the software can mimic the external hardware trigger. See Software triggers for more information.

4.3.3 Software triggers

The software trigger interfaces are provided to enable the software to interact with triggering. They can be used for synchronizing the DMAC operation with the software, or during development, it can be a substitute for hardware trigger events.

To use software triggers, each of the following fields can be configured as either hardware or software type trigger the CH_SRCTRIGINCFG.SRCTRIGINTYP, CH_DESTRIGINCFG.DESTRIGINTYPE and CH_TRIGOUTCFG.TRIGOUTTYPE fields must be configured accordingly.

Software Trigger Interface

The software Trigger Interface consists of the following DMA channel register fields.

The software trigger control fields in the CH_CMD register

- SRCSWTRIGINREQ
- SRCSWTRIGINTYPE
- DESSWTRIGINREQ
- DESSWTRIGINTYPE
- SWTRIGOUTACK

The trigger status fields in the CH_STATUS register

- STAT_SRCTRIGINWAIT

- STAT_DESTRIGINWAIT
- STAT_TRIGOUTACKWAIT

The trigger interrupt status fields in the CH_STATUS register

- INTR_SRCTRIGINWAIT
- INTR_DESTRIGINWAIT
- INTR_TRIGOUTACKWAIT

The trigger interrupt enables fields in the CH_INTREN register

- INTREN_SRCTRIGINWAIT
- INTREN_DESTRIGINWAIT
- INTREN_TRIGOUTACKWAIT

Software trigger protocol

The wait status fields (STAT_*WAIT) indicate when the DMA channel can accept a software trigger. Software triggers cannot be initiated when the wait status field is zero.

Software trigger input protocol

The intended trigger input type must be set in a CH_CMD.*SWTRIGINTYPE field . The trigger input request is initiated by writing 1 into the *SWTRIGINREQ field. This can be done with the same register Write-Access as the one that sets the type or with a consequent Write-Access.

When the request is initiated, the associated STAT_*TRIGINWAIT flag is cleared to indicate that the processing of the request has been begun. The CH_STATUS.*SWTRIGINTYPE field becomes read-only to keep the trigger input type stable throughout the trigger event.

When the trigger request is completed, the *SWTRIGINREQ field is cleared and *SWTRIGINTYPE field changes to be read/write again.

Software trigger output protocol

The CH_STATUS.STAT_TRIGOUTACKWAIT field indicates that a trigger output request is active and the software might send an acknowledge to it bywriting 1 into SWTRIGOUTACK field.

The STAT_TRIGOUTACKWAIT status field is cleared when the acknowledge is accepted.

Software trigger interrupts

The CH_STATUS.INTR_* interrupt status fields are set whenever the corresponding STAT_* fields are set and the corresponding interrupt is enabled by CH_INTREN.INTREN_* being set to 1.

The interrupt flags are cleared automatically when the status flags are cleared.

Related information

- Trigger Interfaces
- Interrupt operation

4.4 Command linking

The command linking feature enables the DMA channels to execute more operations by automatically loading the next commands from the system memory to its configuration registers. This feature makes the DMAC versatile in combining multiple commands in a DMAC transaction.

Each command can define new transfer parameters, triggering behavior, and interrupt settings. This provides great flexibility in the possible usage of the DMA unit and makes it possible to implement complex data transfer tasks by properly designing command chains. The commands are defined by descriptors stored in memory.

The channel fetches the descriptors by using the pointer address stored in the link register CH_LINKADDR. The first command must be set directly in the channel registers to start the linked command chain. If required, the first command can be an empty command which only points to a memory location with the first non-empty command.

Certain commands might only change some of the configuration registers, for example, the destination address or the number of transfers, but keep all other registers the same for the next command. To do this, the command descriptors in the memory have a header part that specifies what registers to update in the channel register bank. This reduces the number of bus transfers executed during command fetching.

The command link chain finishes when a command in the chain does not have the LINKADDREN bit set. This can be achieved by not setting this LINKADDR register bit in the descriptor of the last command in the chain.

The new command is fetched after the DONE state when a command is finished, see Channel lifecycle. The register values are read and updated according to the header word.

Execution of a command chain can be stopped by setting the DISABLECMD bit in the channel command register CH_CMD. It can be used to stop an endless looped command link. The disable stops the command chain cleanly; the command being executed currently is finished but the next one is not loaded.

When a command in a command link is paused, it behaves just like pausing a single command, see Stop and pause control. After resuming, the execution of the command is resumed and the command link is continued.

When a command in the command link stopped, the remaining part of the command and the remaining part of the command link are canceled. The stop waits for all the outstanding responses from read and write transactions but it tries to finish the channel operation as soon as possible. For more information, see Stop and pause control.

4.4.1 Command structure

The commands are stored in the system memory within a linked list data structure. Each command has information on what DMA channel configuration registers must be updated and a pointer to the next command.

The pointer to the next command is stored in the CH_LINKADDR configuration registers. The registers can be updated by the next command as any other channel configuration register.

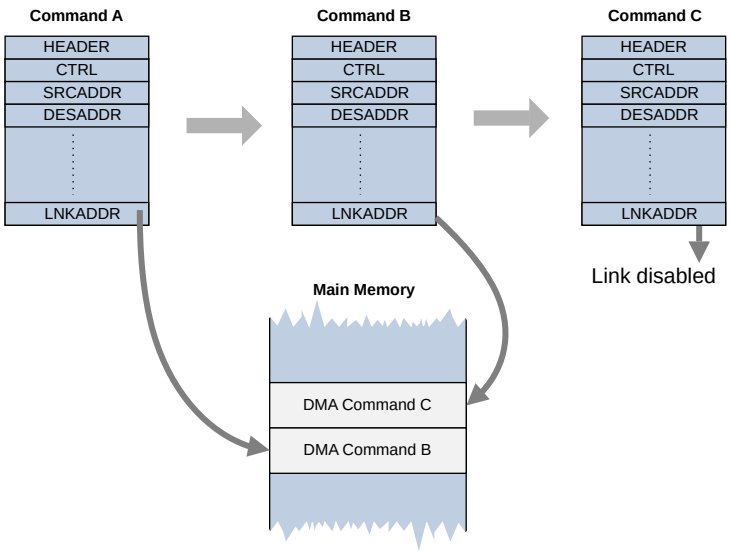
The command chain is terminated by disabling the CH_LINKADDR.LINKADDREN flag. All other commands must have this flag enabled to indicate a valid link address.

The first command of the command chain must be set manually in the channel configuration registers. The first command can be empty and only link to the first real command in the chain.

Linked command chain structure

The following figure illustrates a command chain data structure:

Figure 4-12: Command linking operation



Command descriptors

The commands descriptors are encoded within a continuous array of 32-bit words in the memory.

The first 32-bit word of each command serves as a header that contains a bitmap to indicate what registers require updating. The header also has a special flag, REGCLEAR, that instructs the DMA channel to clear all the configuration registers that are modifiable by command link before updating them with the new values of the command. If this flag is not set, the configuration registers not updated by the command are kept at their previous settings.

The following table specifies the header word of the commands:

Table 4-4: Command link header

Bit	Name	Description
[0] (LSB)	REGCLEAR	Clear registers before updating with new values
[1]	Reserved	Reserved for future use, must be zero
[2]	INTREN	Update the CH_INTREN register
[3]	CTRL	Update the CH_CTRL register
[4]	SRCADDR	Update the CH_SRCADDR register
[5]	Reserved	Reserved for future use, must be zero
[6]	DESADDR	Update the CH_SRCADDR register
[7]	Reserved	Reserved for future use, must be zero
[8]	XSIZE	Update the CH_XSIZE register
[9]	Reserved	Reserved for future use, must be zero
[10]	SRCTRANSCFG	Update the CH_SRCTRANSCFG register
[11]	DESTRANSCFG	Update the CH_DESTRANSCFG register
[12]	XADDRINC	Update the CH_XADDRINC register
[13:18]	Reserved	Reserved for future use, must be zero
[19]	SRCTRIGINCFG	Update the CH_SRCTRIGINCFG register
[20]	DESTRIGINCFG	Update the CH_DESTRIGINCFG register
[21]	TRIGOUTCFG	Update the CH_TRIGOUTCFG register
[22]	GPOEN0	Update the CH_GPOEN0 register
[23]	Reserved	Reserved for future use, must be zero
[24]	GPOVAL0	Update the CH_GPOVAL0 register
[25:27]	Reserved	Reserved for future use, must be zero
[28]	LINKATTR	Update the CH_LINKATTR register
[29]	AUTOCFG	Update the CH_AUTOCFG register
[30]	LINKADDR	Update the CH_LINKADDR register
[31] (MSB)	Reserved	Reserved for future use, must be zero


Note

At least one bit of the header word must be set to 1, otherwise a configuration error is reported and execution of the command stops. For more information, see Error handling.

The consecutive words in a command must contain the new values of the configuration registers that were indicated for updating in the header. They must be in the same order as the flags in the header starting from LSB to MSB.

Example descriptors for linked commands

An example of three descriptors in the memory when 32-bit addressing is used:

Table 4-5: Command link example memory map, HEADER_0

Descriptor offset	Field	Value
0x000	HEADER_0	0x0000_0D5D
0x004	INTREN	Any
0x008	CTRL	Simple 32-bit wide 1D
0x00C	SRCADDR	Any
0x010	DESADDR	Any
0x014	XSIZE	Any
0x018	SRCTRANSCFG	Any
0x01C	DESTRANSCFG	Any

Table 4-6: Command link example memory map, HEADER_1

Descriptor Offset	Field	Value
0x020	HEADER_1	0x4000_0158
0x024	CTRL	Simple 8-bit wide 1D
0x028	SRCADDR	Any
0x02C	DESADDR	Any
0x030	XSIZE	Any
0x034	LINKADDR	Descriptor offset + 0x038 + 1 (ENABLE BIT)

Table 4-7: Command link example memory map, HEADER_2

Descriptor Offset	Field	Value
0x038	HEADER_2	0x4000_0140
0x03C	DESADDR	Any
0x040	XSIZE	Any
0x044	LINKADDR	0x000

The Any value means that the software can set it to whatever the command requires.

The HEADER_0 defines a command that sets up a complete 1D transfer with all registers necessary. This is because the header contained the REGCLEAR bit set which clears all previous settings from the channel registers. It also terminates that command chain because the REGCLEAR command cleared the LINKADDR as well.

Another chained command is stored in the memory defined by HEADER_1, independent from the first one. It sets a new CTRL register content, updates SRCADDR and DESADDR values and sets the XSIZE to define the length of the command. The other register settings remain the same as the first command in this command chain. It also updates the LINKADDR register and links the command to the next command defined by HEADER_2. When the command defined by HEADER_1 is finished, the next descriptor at HEADER_2 is read. This only changes the DESADDR and the XSIZE compared to the previous command. This might be used to copy the trailing source data to a different location. This is the final command in the chain, so it terminates the link by setting the LINKADDR register to 0 that clears the LINKADDREN enable bit.

4.4.2 Loading commands

The command linking feature loads commands from the system memory using the same Data AHB5 bus manager interfaces that are used for data transferring.

Command descriptor fetch attributes

The command is fetched from the memory with AHB5 burst reads with TRANSIZE parameter set as the value of the DATA_WIDTH parameter.


Note

The value of the CH_SRCTRANSCFG.SRCMAXBURSTLEN field does not apply to command descriptor reads.

The command link read uses the security and privilege state of the channel regardless of the transfer properties set in the CH_SRCTRANSCFG or CH_DESTRANSCFG registers. As a result, a Non-secure channel can only load Non-secure commands and a Secure channel can only load Secure commands.

Command link reads are distinguished on the AHB5 bus from data transfers with hprot_dmaif[0] set to 0 (instruction access).

Updating configuration registers

The header word is the first word that is read as part of the command link fetch. The consecutive reads contain the new values of the configuration registers that are updated. The register values are updated in the order as they are received.

4.4.3 Automatic boot feature

The DMA unit implements an automatic boot feature called autoboot to speed up the bootup of the system. The autoboot can load the first DMA command into channel 0 and start executing it. Autoboot is implemented with the command-link feature and it can be configured with dedicated input signals.

The boot command or commands must be prepared in memory, which may contain any DMAC configuration register value supported by command linking. This provides the possibility to set up an initial single memory-to-memory copy command or a chain of multiple commands that are executed automatically at start-up.

Enabling the autoboot feature

The address of the boot command and some additional attributes must be configured as input signals, and the boot enable signal must be set HIGH to enable the autoboot feature. To execute an autoboot, the following signals must be set before deactivating the device reset. They must also be stable until the fetching of the boot command is started.

- boot_en = `0b1`

- boot_addr = <address of boot command descriptor>
- boot_priv = <privilege status setting for Channel 0>
- boot_memattr = <AHB5 memory attributes to be used during fetching the boot command descriptor>
- boot_shareattr = <AHB5 memory Shareability attributes to be used during fetching the boot command descriptor>

When Security Extension is used, DMA channel 0 starts as Secure, therefore the boot address must point to a Secure address location.

Autoboot process

The autoboot process is initiated after DMA-250 is released from reset and both LPI channels have entered their active state. The input signals are sampled at this point, and the process begins with writing the preconfigured boot address and attributes to Channel 0 configuration registers as an initial, empty linked-command.

After this process, the boot command referenced by boot_addr is fetched and gets written to Channel 0 configuration registers. Finally, the DMAC executes the boot command.

Channel 0 status signals can be used to track the boot process. For example, ch_enabled[0] is asserted throughout the boot command like any other DMA command executed on Channel 0.

4.5 Channel arbitration for AHB5

Channel arbitration (in this context, an allocated VCH and PCH pair) is required when multiple physical channels and other DMA control requestors want to use a single bus interface to send transfers.

These transfers can come from DMA command data reads and writes, context fetch and save, command link reads, APB5 accesses to registers in the context memory or clearing the registers with their default values in the context memory.

Each virtual channel has a register to set the channel's priority and this value is inherited by the allocated physical channel. The priority register defined in the architecture affects the arbitration between virtual channels in the context manager, and the data AHB and context AHB bus arbitration between physical channels. The software is responsible for adjusting the priorities of the channels properly as the DMA Unit does not know which channel operation is more time critical than the other.


Note

Software can change the priority level of a channel by:

- Writing to the CHTRL.CHPRIO field through APB5 when the (virtual) channel is disabled.
- Updating CHTRL.CHPRIO field by the command descriptor during command linking (bit[3] in the command link header must be set).

4.5.1 LRG Arbitration scheme

The Data BIU and the Context BIU use the same arbitration policy.

The priorities are set in CH_CTRL.CHPRIO by software and can only be changed when the channel is in IDLE state.

The requestor priorities of the register clear logic and the APB5 to Context memory bridge logic are handled dynamically so that they are always match the priority level of the highest priority channel in the arbitration.

For details of the arbitration scheme, see Arbitration policy.

In some corner cases, starvation can occur for lower priority channels when high priority requests and low-priority requests are interleaved. For example, this starvation can be caused by an infinitely looped command linked list where the software frequently enables a higher priority channel and takes all the timeslots from lower priority channels. Reads for command linking also happen with the same priority and same arbitration scheme as the other data-related reads.

These cases can be avoided by managing the risks of starvation in software when using high priority channels together with low-priority ones. The DMAC does not implement fair share algorithms to avoid these situations and it is up to the software designer to take this into consideration.

In a generic approach, free timeslots occur when the high priority channels wait for trigger, wait for responses to arrive, get reconfigured over APB5 or read a new command from the linked list. These free timeslots could be used to enable the low-priority channels to progress in their operation.

Infinitely looped command link lists can result in starvation of lower priority levels.

4.6 DMAC Power management

DMA-250 has separate power and clock control management systems.

4.6.1 Power P-Channel

DMA-250 uses one full LPI P-Channel for power management. The purpose of this feature is to enable lower power states (removing power) to reduce power consumption while not in active use. The power P-Channel is used to control the power state of the DMAC logic.

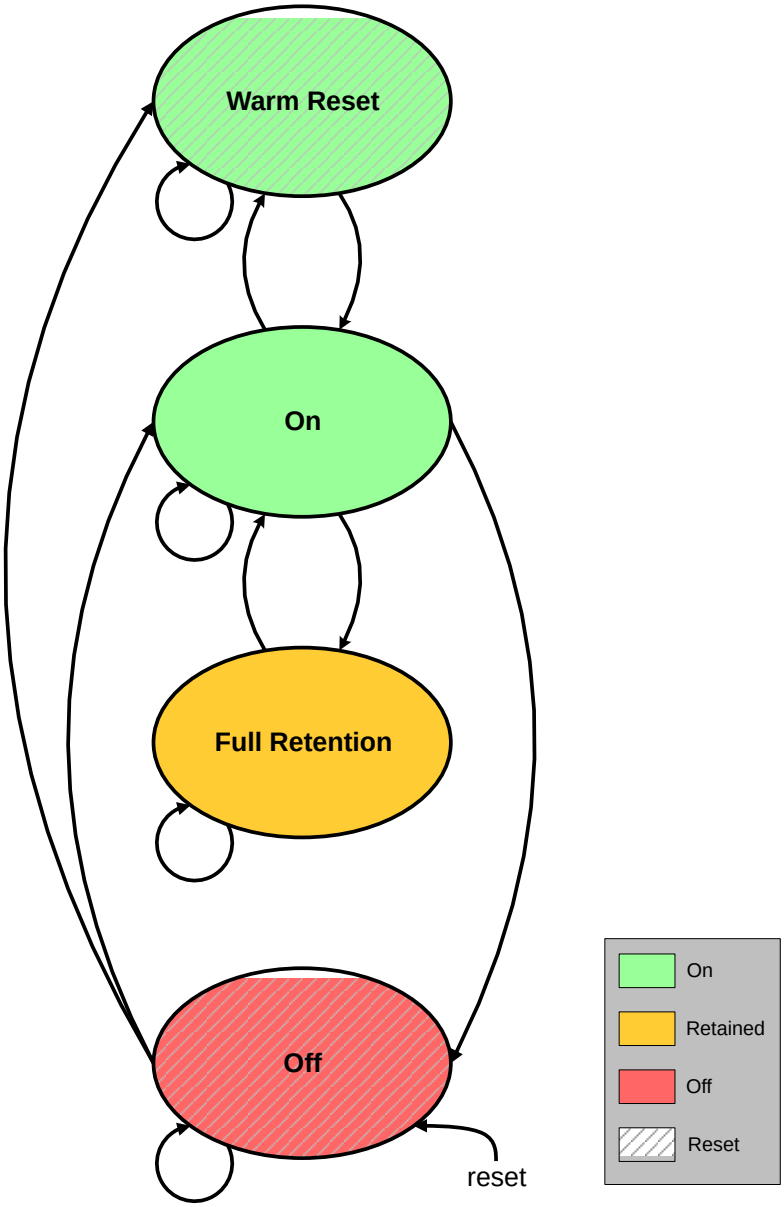
The following four power states can be requested over the P-Channel interface, any other state requests are denied:

- On
- Warm reset mode
- Full retention mode
- Off

The DMAC has a simple power management scenario with the P-Channel interface. When the DMAC is active, power quiescence requests are denied and the operation simply continues. Warm reset mode request pauses all ongoing channel operation and exiting from Warm reset mode to On resumes the channel operation. When the DMAC is actively waiting for an event, the quiescence request to Full retention mode state is accepted but Off state is denied. When the DMAC is inactive, request to Off state is accepted. Before entering Off state, FIFOs are emptied and register values are not kept.

The following figure shows the supported power states and transitions between power states.

Figure 4-13: Supported power states and transitions



Entering lower power modes (Full retention mode, Off) can be disabled with the DISMINPWR field of configuration registers NSEC_CTRL and SEC_CTRL. In effect of the settings of the registers the power state request is denied if the minimum power state set in DISMINPWR is higher than the requested power state and there is at least one channel configured as Non-secure or Secure respectively.

Entering Full retention mode state while the DMA is in an actively waiting state can be Disabled and Enabled by the IDLERETEN field of configuration registers NSEC_CTRL and SEC_CTRL. In effect of the settings of these registers the power state request is always denied if IDLERETEN is set to Disabled, the requested state is Full retention mode and at least one Non-secure or Secure channel (according to the IDLERETEN field's security) is waiting for an event.

For more information on the P-Channel handshake mechanism, see the *AMBA® Low Power Interface Specification* document.

Related information

- LPI power P-Channel
- P-Channel interface signals

4.6.2 Clock Q-Channel

DMA-250 uses one full LPI Q-Channel for clock management. The purpose of this feature is to reduce power consumption by shutting down the clock while not in active use. The clock Q-Channel is used to turn on and off the clock for the DMAC logic.

The Q-Channel interface controls the DMAC clock management. When the DMAC is active, clock quiescence requests are denied and the operation simply continues. When the DMAC is actively waiting for an event or is inactive, the quiescence request is accepted and the clock can be shut down.

For more information on the Q-Channel handshake mechanism, see the *AMBA® Low Power Interface Specification* document.

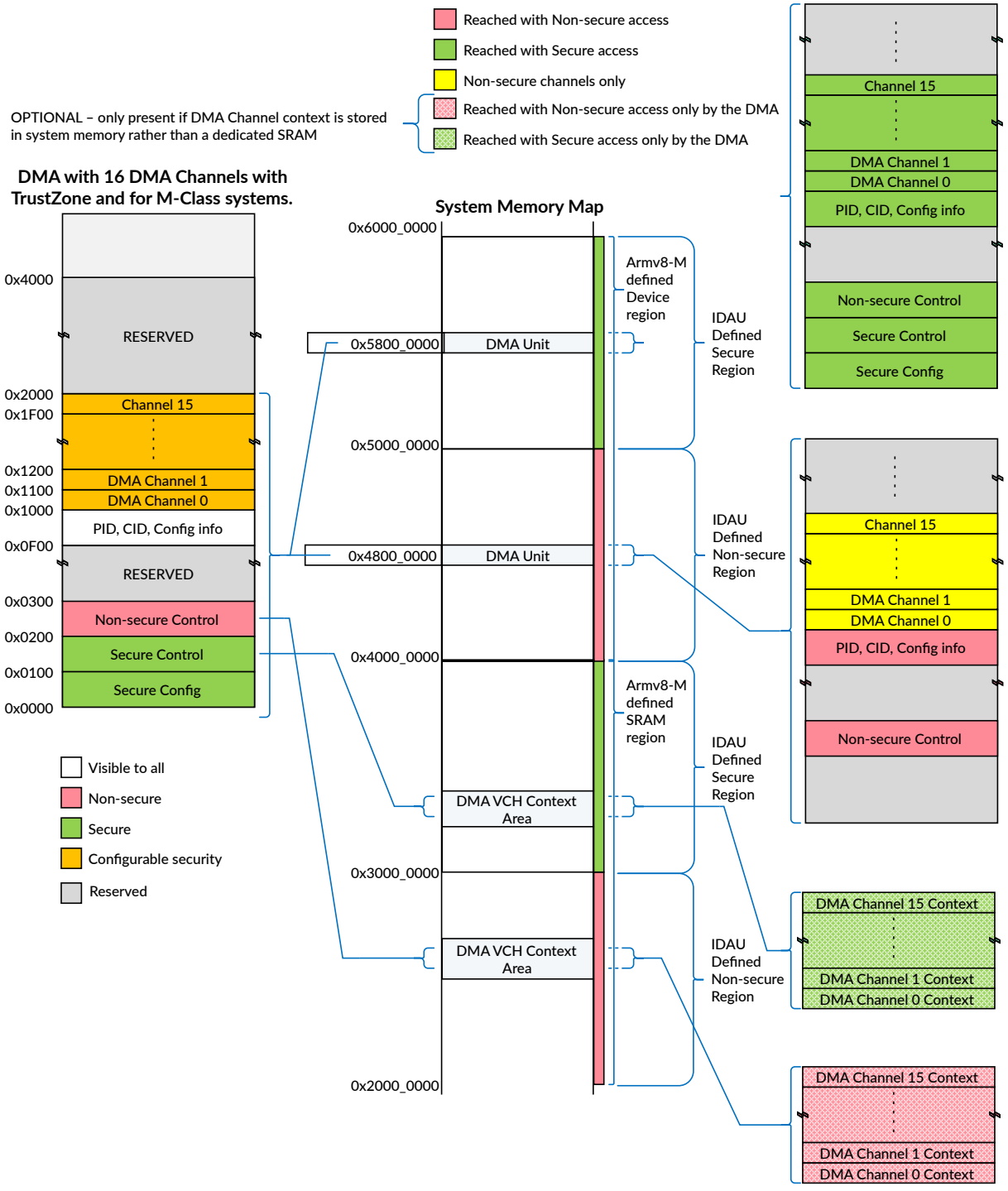
For Q-Channel interface signals, see Signal descriptions.

4.7 Configuration

DMA-250 can be set up through its configuration registers. These registers are divided into several frames. Each register frame occupies a 256 byte address space.

The layout of the register frames is shown in the following figure.

Figure 4-14: DMA register frame layout



The whole DMA unit occupies an 8kB configuration address space.

DMA unit configuration

The following four register frames are associated with the DMA unit:

Security Configuration Register FrameDMASECCFG

This frame contains registers to configure the security and privilege of each DMA channel, and the security of the external trigger ports.

Secure Control Register Frame, DMASECCTRL

This frame contains status and control registers that affect all channels configured as Secure.

Non-secure Control Register Frame, DMANSECCTRL

This frame contains status and control registers that affect all channels configured as Non-secure.

Information Register Frame, DMAINFO

This frame provides information about the capabilities and parameters of the DMA unit. For details of unit configuration, see DMAINFO summary.

4.7.1 Channel configuration

Each DMA channel has its own register frame which contains the channel-specific configuration registers. The configuration registers are writable only when the channel is not enabled. When the ENABLECMD bit is set in the CH_CMD register, the settings are frozen throughout the execution of the command.

To configure a command correctly, the settings written to the configuration registers must meet the following criteria:

- The register fields must contain allowed values only. To see which values are valid, see Programmers model. All bits of reserved fields must be zeroes. Failing to meet this criteria results in REGVALERR.
- The settings must describe a command that is valid, that is, there are no conflicting parameters. Failing to meet this criteria results in a onfiguration error (CFGCONFLERR). For more details, see Error handling.

To configure a DMA Channel command, see Step 4 Configuring a DMA Channel command.

Empty commands

There is a subset of settings that describe a valid command, but the execution does not result in actual data transfers. Such commands, called empty commands, can be useful. For example, to initiate a linked command chain, to set a GPO value, or to synchronize with other entities using the triggering features.

An empty command is executed if the CH_CTRL field is explicitly set to ‘disable’ `0b000`, or when neither data reads nor data writes are configured for the AHB5 interface.

The following cases result in no data reads through the Data AHB5 interface:

- Explicit empty command is requested:

```
XTYPE = 'disable'
```

- Data source size (and therefore destination size) is set to zero:

```
(SRCXSIZE = DESXSIZE = 0)
```


Note

The recommended method for configuring an empty command intentionally is to set XTYPE to 'disable'.

Configuration errors because of invalid settings

The following invalid settings result in configuration errors. No data transfers are made, and the CFGERR and REGVALERR fields are set in the CH_ERRINFO register.

- CH_CTRL.TRANSIZE is set as greater than the bus width.

Configuration errors because of conflicting settings

The following settings result in configuration conflict errors. No data transfers are made, and the CFGERR and CFGCONFLERR fields are set in the CH_ERRINFO register.

- Flow control trigger modes imply that there must be data transfer configured for the given direction. For the definitions of no AHB5 read or write configured, see the Empty commands section above.

  Therefore, the following settings are illegal:

  - Source flow control trigger input mode when no AHB5 source data:

```
USESRCTRIGIN = 0b1 and SRCTRIGINMODE = 20b1X and No AHB5 read configured
```

  - Destination flow control trigger input mode when no AHB5 destination data:

```
USEDESTRIGIN = 0b1 and DESTRIGINMODE = 20b1X and No AHB5 write configured
```

Related information

- DMACH<n> summary, DMA Channel Register Frame

4.7.2 Halting and restarting the DMA with Cross Trigger Interface

DMA-250 can be halted for debug purposes using the Cross Trigger Interface (CTI) interface.

When a HIGH is received on the CTI halt_req signal and the DBGHALTEN bits in the NSEC_CTRL and SEC_CTRL registers are set, pausing of both the Secure and Non-secure channels is initiated.

When all the channels reach either paused or not enabled state after a halt request, a pulse on the halted CTI signal is sent. Meanwhile, the DMAC monitors the restart_req CTI signal, regardless

of whether all the channels have reached their paused or disabled state. When a rising edge on restart_req is received, or if SEC_CTRL.DBGHALTEN register field is cleared, the DMA channels return to their state before the halt request.

The CTI halt request and restart request are ignored in certain cases:

- The halt_req is ignored after a previous halt request and before the DMAC returns to normal operation.
- The restart request is ignored before a halt request is received. After a valid HIGH is received on the restart_req, further level changes are ignored.

The halt request from the CTI is combined with the hardware pause, allch_pause_req, and the software initiated allchpause. The DMA channels can be paused using any of the three methods.

The following waveform and Table 4-8: Stages of normal CTI interface operation on page 74 illustrate the normal behavior of the interface.

Figure 4-15: CTI interface normal operation

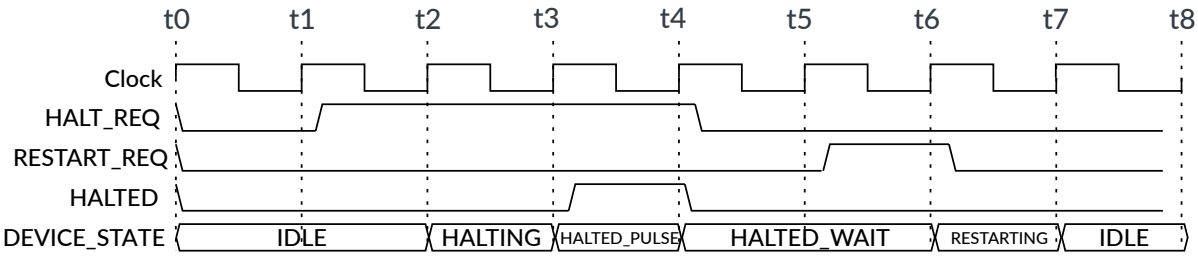


Table 4-8: Stages of normal CTI interface operation

Stage	Description
t1	The halt request initiator sets the level HIGH.
t2	The DMA detects the HIGH level of the request and starts the halting process. This might take multiple cycles. The request is kept HIGH but the DMA ignores its value. The request is allowed to be deasserted at any time.
t3	The DMA finished the halting process and asserts the 1-cycle pulse on the halted status signal.
t4	The DMA remains in this state and starts to look for the HIGH pulse on the restart request.
t5	The debugger sends a restart request pulse to continue the operation.
t6	The DMA detects the HIGH value on the restart request and starts the restarting process. This might take multiple cycles. The DMA is already capable to accept a new halt request at this point.
t7	The DMA continues its operation.

Related information

- Cross Trigger Interface
- Table A-11: Cross Trigger Interface signals on page 170

4.8 Initialization

The DMAC must be configured to set the security and privilege attributes for each channel, their attached resources before setting the channels for memory transfers, and to input the initial base address of the Context memory area. When the configuration is complete, the channel configuration registers are protected against malicious accesses.The security configuration can be locked until the next reset of the DMAC to further protect the security settings.

Privilege settings of a channel can be adjusted when the channel is in IDLE. The register settings are automatically cleared when the security or privilege settings of a channel change.


Note

We recommend that the security or privilege change is done with a sequence of writing and reading back the desired new state to assure the success of the change.

After reset, when both LPI channels have entered their active state:

1. The DMAC checks the boot_en port:
   - If automatic booting is turned on, the DMAC sets the boot address as the first link address for channel 0 and enables the channel before any APB5 configuration can occur.
   - When security is enabled for the DMAC, the channel 0 is set to be Secure as all other channels so only Secure commands can be fetched from the memory.
2. The DMAC checks the initial_cntxbase[31:12] signal to the Context memory and sets the relevent register as the base address:
   - The SEC_CNTXBASE register, if TrustZone support is enabled
   - The NSEC_CNTXBASE register, if TrustZone support is disabled
3. The DMAC samples the initial_cntxmem_clr signal. If it detects a '1', the DMAC initializes the Context memory area of the channels that are not allocated by default:
   - When TrustZone support is enabled, only the secure Context memory area is cleared, as all channels are reset to Secure. If software changes the security of one or more channel, the DMAC initializes their Non-secure Context memory area.
   - When TrustZone support is disabled, the Context memory area is cleared.


Note

For the DMA to operate properly, the Context memory must be initialized after a cold reset. The length of this process depends on the NUM_VCH parameter, and for large configurations this may be a considerable amount of time. However, depending on your system, initialization may be omitted after a warm reset if the context memory area retains its contents.

In such a system, the initial_cntxmem_clr signal can be driven from a system control register that resides outside of the DMA's reset domain and which is aware of the type of reset that was applied. To keep the implementation simple, Arm recommends connecting the initial_cntxmem_clr signal to logic 1.

Related information

- Channel arbitration for AHB5
- SEC_CNTXBASE
- NSEC_CNTXBASE
- Configuration signals

4.9 Interrupt operation

Interrupts provide indication of internal state changes of the DMA channel and of the DMA unit.

Each channel has its own separate interrupt, but there are global DMA level interrupt signals that show security errors or global operational state changes. One unit level Non-secure interrupt is always present and in addition to that, one Secure unit level and one Secure violation interrupt also appears when Trustzone support is enabled (SECEXT_PRESENT=1).

Interrupts are level based, the enable and clear operations are software controlled. When the security extension is enabled, Secure and Non-secure interrupts are used. Each interrupt has a handler that is running in the proper security world, so alignment is needed when changing the security state of the channels. The channel interrupt is considered Secure when the channel is configured as Secure. Otherwise, only the Non-secure counterparts are implemented.

These interrupts combine the state of the individual Secure channel interrupts (irq_comb_sec) and the individual Non-secure channel interrupts (irq_comb_nonsec). The SEC_CHINTRSTATUS0 register for the Secure channels and NSEC_CHIRQSTATUS0 register for the Non-secure channels hold the combined channel interrupt statuses. When changing the security state of the channels, alignment is needed to map the corresponding channel interrupt from one world to the other. The combined interrupts also include the interrupt statuses for all channel idleness. These statuses are generated in the control block and can be read and cleared in the SEC_STATUS and NSEC_STATUS registers.

All interrupt requests are active-HIGH level based. The enable and clear operations are software controlled via register writes over the APB5 completer interface.

Table 4-9: Non-secure DMA level interrupt signal sources

Interrupt source	Description
nsec_chintrstatus0	Collated channel Non-secure interrupts register.
intr_anychintr	This interrupt is a combined Non-secure interrupt, combining the other Non-secure interrupt sources and all the Non-secure channel interrupts when NSEC_CTRL.INTREN_ANYCHINTR is set to 1.
intr_allchidle	This interrupt is raised when every Non-secure channel returns to IDLE state from a non-IDLE state. The interrupt is not asserted after reset.
intr_allchstopped	This interrupt is raised when the last Non-secure channel enters stopped state.
intr_allchpaused	This interrupt is raised when the last Non-secure channel enters paused state.
intr_ctxerr	This interrupt is raised when accessing a Non-secure channel's context memory area receives an error response.

Table 4-10: Secure DMA level interrupt signal sources

Interrupt source	Description
secchintrstatus0	Collated channel Secure interrupt flags.
intr_anychintr	This interrupt is a combined Secure interrupt, combining the other Secure interrupt sources and all the Secure channel interrupts when SEC_CTRL.INTREN_ANYCHINTR is set to 1.
intr_allchidle	This interrupt is raised when every Secure channel returns to idle state from a non-idle state. The interrupt is not asserted after reset.
intr_allchstopped	This interrupt is raised when the last Secure channel enters stopped state.
intr_allchpaused	This interrupt is raised when the last Secure channel enters paused state.
intr_ctxerr	This interrupt is raised when accessing a Secure channel's context memory area receives an error response.

The irq_sec_viol_err interrupt signal is provided to notify the Secure entity that a security violation has occurred with register access to the DMA. The irq_sec_viol_err interrupt signal only exists when SECEXT_PRESENT is set to 1.

Related information

- Software trigger interrupts
- SEC_CHINTRSTATUS0
- NSEC_CHINTRSTATUS0
- SEC_CTRL
- NSEC_CTRL
- Control interrupts

4.9.1 DMA Control block

The DMA control block defines the operation of the DMA Unit globally and handles the channel related security filtering and other restrictions.

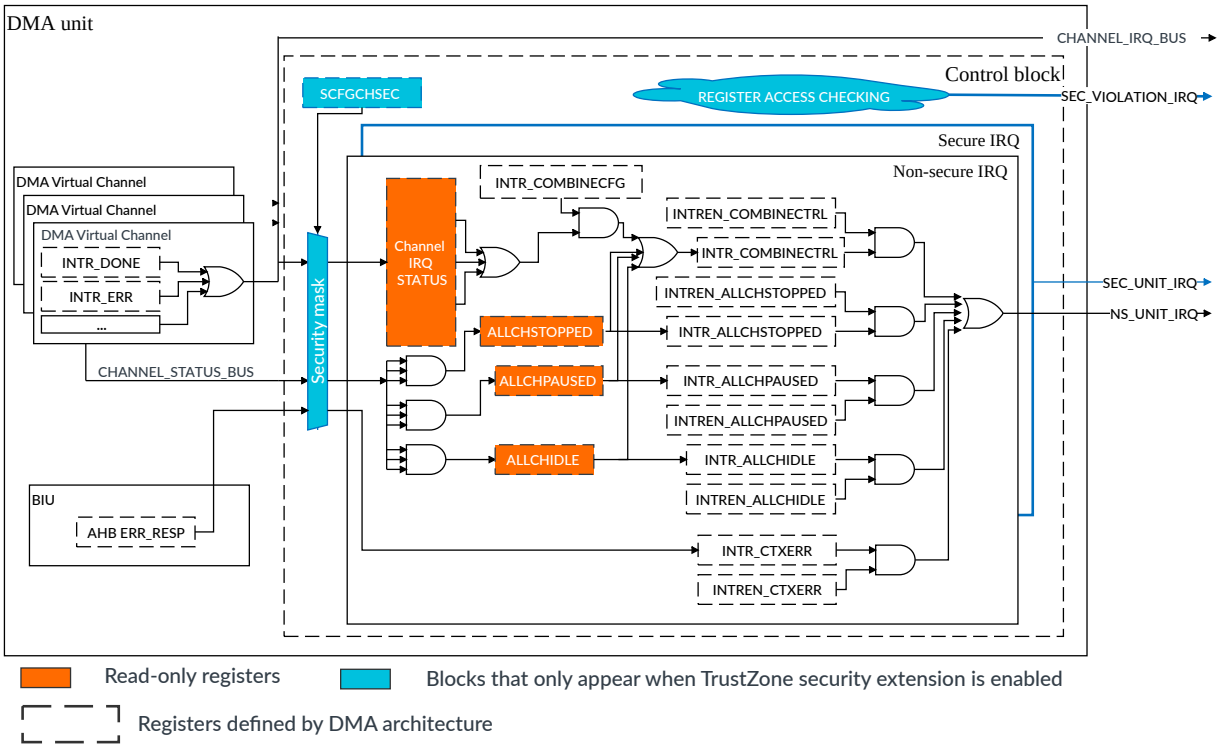
The control block has its own register bank that can be accessed by privileged software via a separate address range. This is the DMA Unit Non-secure Control Register Frame.

When TrustZone™ is enabled (SECEXT_PRESENT = 1), there is another register bank, the DMA Unit Secure Control Register Frame. This is accessible only by Secure privileged software.

Control interrupts

The channel and unit level interrupts can be seen in the following figure.

Figure 4-16: DMA Unit level interrupt wiring



The channel interrupts and status signals are routed to the control block and go through a security mask that selects the interrupts and status bits that belong to the Secure or Non-secure world. The interrupts are only shown in the control block as pending interrupts if they are enabled on channel level. The control block can combine these with other status related interrupts to generate a Non-secure and a Secure unit level interrupt. This can be used for higher level DMA control.

When the security extension is enabled, the control block provides register access checking and can generate a security violation interrupt to indicate a faulty access.

For details of interrupt operation, see Interrupt operation.

The relevant channel level registers are CH_STATUS and CH_INTREN.

The relevant unit level registers are SEC/NSEC_STATUS and SEC/NSEC_CTRL registers, and SCFG_INTRSTATUS and SCFG_CTRL registers.

Control configuration

The control block provides unit level configuration settings that define the behaviour of the whole DMA.

All-channel stop and pause

Both Secure and Non-secure worlds can initiate an all-channel stop and pause request to disable the operation of the channels that belong to that world.

Power management

The Control block also provides settings for power management of the DMA. This can be used to allow keeping the DMA's power domain up even if all channels are idle. The DMA also can be configured to allow/disallow retention when the channels are enabled but in wait for trigger state.

Channel settings

The control block also provides channel related settings that can be adjusted for every channel that belong to that security world. The channels can have unique channel identifiers that allow matching the channels and their AHB5 accesses to a software process. The channels can be selected to run in privileged mode but this needs to be adjusted by a privileged entity. That is the reason these settings cannot be part of the channel register bank.

Control APB5 selection

The Control block contains an APB5 multiplexer that decodes the internal PSEL signals from the address of each channel. It also provides security filtering and response generation when accesses are not allowed to pass to the channel. When a channel is enabled the majority of the registers are in read-only mode from software. The control block filters write accesses to all channels based on their status.

When a channel is in Enabled state, then all channel registers become read-only except for the CH_CMD and CH_STATUS registers that enable general control of the channel operation and the clearing of interrupts. Write accesses to other registers are ignored.

Channel registers include registers in the DMA Channel Register Frame and the CHID field of SEC_CHCFG register in the DMA Unit Non-secure Control Register Frame and DMA Unit Secure Control Register Frame.

Table 4-11: APB5 Security filtering rules according to access type

Register Type	Non-secure, unprivileged PPROT[1:0] = 0b10	Non-secure, unprivileged PPROT[1:0] = 0b11	Secure, unprivileged PPROT[1:0] = 0b00	Secure, unprivileged PPROT[1:0] = 0b01
Non-secure, unprivileged channel	Allowed	Allowed	Allowed	Allowed
Non-secure, privileged channel	**RAZ/WI**	Allowed	**RAZ/WI**	Allowed
Secure, unprivileged channel	**RAZ/WI** or Error interrupt	**RAZ/WI** or Error interrupt	Allowed	Allowed
Secure, privileged channel	**RAZ/WI** or Error interrupt	**RAZ/WI** or Error interrupt	**RAZ/WI**	Allowed
Secure privileged unit register frame DMANSECCTRL	**RAZ/WI**	Allowed	**RAZ/WI**	Allowed
Secure privileged unit register frame DMASECCTRL	**RAZ/WI** or Error interrupt	**RAZ/WI** or Error interrupt	**RAZ/WI**	Allowed
Security config register frame DMASECCFG	**RAZ/WI** or Error interrupt	**RAZ/WI** or Error interrupt	**RAZ/WI**	Allowed
DMAINFO register frame DMAINFO	Allowed	Allowed	Allowed	Allowed
Reserved memory space	**RAZ/WI**	**RAZ/WI**	**RAZ/WI**	**RAZ/WI**

APB5 access routing

APB5 register access is either:

- Served directly from within the DMA
- In some cases the DMA generates an AHB5 transfer on its Context AHB5 interface, and the response it gets on AHB5 determines the response it gives on APB5.

  This access routing occurs when the APB5 access targets the PCH registers of a non-allocated channel.

4.10 Register clear feature

The DMAC implements a housekeeping mechanism that clears all registers in the DMACH register frame, and when present, the CHID register. If a VCH is deallocated when register clear is performed on the channel's registers, the DMAC clears the corresponding area in CTXMEM (Context memory) by generating write transfers on the context AHB5.

The following table lists the events that result in register clearing.

Table 4-12: Register clearing events

Register Clear Source	Scope	Description
Changing channel privilege setting	Single channel	Changing a channel's privilege level (CHPRIV field of SEC_CHCFG / NSEC_CHCFG register) results in register clear for the channel.
Changing channel security setting	Single channel	Changing a channel's security result in register clear for the channel. If the channel is deallocated, its CTXMEM area in the new security domain is cleared.
CH_CMD.CLEARCMD	Single channel	Setting the CLEARCMD bit in the channel's CH_CMD register results in register clear for the channel when its DMA operation is either completed, disabled or stopped. **Note:** Care should be taken when using the CH_CMD.CLEARCMD feature as it comes into effect at the end of the command, regardless of whether the command finished normally or if it was interrupted during operation by an error or STOP request. Since CLEARCMD clears the DMACH registers, debugging interrupted commands that use CLEARCMD might be more difficult.
Changing CTXMEM base address	All channels	Writing the CTXMEM base address (SEC_CNTXBASE / NSEC_CNTXBASE registers) results in register clear for all channels in the given security domain.
CTXMEM initialization after reset	All channels	When the DMAC samples the initial_cntxmem_clr as `0b1` after reset when power and clock become active, the DMAC performs register clear for all channels. This also involves the CTXMEM area of the channels that are not allocated by default. When TrustZone support is enabled, only the secure CTXMEM area is cleared, as all channels are reset to secure. If software changes one or more channel's security, the DMAC initialize their Non-secure CTXMEM area.

Related information

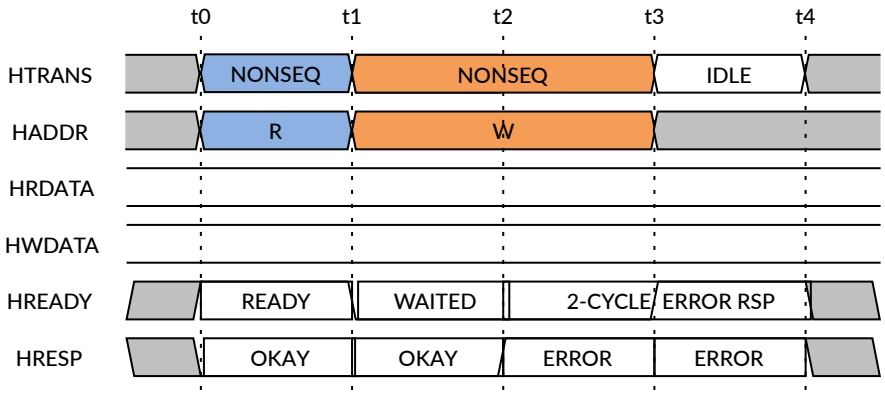
- DMACH<n> summary, DMA Channel Register Frame

4.11 Error response

During DMA operations, if a Data-AHB5 transfer receives an error response, then the corresponding DMA channel cancels any transfers remaining from the command, and the channel stops operation. Then in the CH_STATUS register, the STAT_STOPPED and STAT_ERR flags are set.

The following figure depicts a waited data read transfer that is responded with the two-cycle AHB5 error response, and so the DMA channel cancels the following write transfer by driving HTRANS to IDLE. This is in accordance with AHB5 specification.

Figure 4-17: A data read responded with error cancels the corresponding data write during DMA operations



This also applies to scenarios when a data write or a command descriptor read transfer receives an error response.

While Data-AHB5 error responses are considered to be specific to the command the channel is executing, an error response on the Context-AHB5 interface affects the DMA itself fundamentally, as without being able to access the context memory the DMA cannot operate properly.

Related information

- CH_STATUS

5. Getting started

Follow a series of steps to set up one or more DMA channels after reset.


Note

The exact values to program into the registers listed below depend on your use case.

For the detailed register descriptions, see Programmers model.

5.1 Step 1 Setting the CNTXBASE addresses

As the first step, you must set the context memory area.

If Security Extension is present

If Security Extension is present in your configuration:

1. The DMA samples the base address of the Secure context memory area from the initial_cntxbase input into the SEC_CNTXBASE register in the DMA Unit Secure Control Register Frame. This sampling takes place once after reset, when power and clock become stable. Secure Privileged software can change this value by programming the SEC_CNTXBASE register.

   Writing to the SEC_CNTXBASE register triggers the DMA's clearing of the Secure context memory area.
2. Secure Privileged software then must program the base address of the Non-secure context memory area by writing the address value to the NSEC_CNTXBASE register in the DMA Unit Non-secure Control Register Frame.

   This step must be performed before assigning any DMA Channels to the Non-secure domain.
3. Non-secure Privileged software can also program the NSEC_CNTXBASE register.

   Writing to the NSEC_CNTXBASE register triggers the DMA's clearing of the Non-secure context memory area.

If Security Extension is not present

The DMA samples the base address of the Non-secure context memory area from the initial_cntxbase input into the NSEC_CNTXBASE register in the DMA Unit Non-secure Control Register Frame once after reset. After reset, the power and clock become stable.

Non-secure Privileged software can change this value by programming the NSEC_CNTXBASE register.


Note

Writing to the NSEC_CNTXBASE register triggers the DMA's clearing of the Non-secure context memory area

5.2 Step 2 Assigning channel security

Each channel must be assigned to either the Secure or to the Non-secure world. Configuring a channel's security determines if Non-secure software can access the channel's registers and also whether the channel can perform Secure transfers. Secure channels can only be accessed by Secure software, and only Secure channels can perform Secure DMA transfers.

If Security Extension is present

If Security Extension is present in your configuration, Secure Privileged software must configure the security of each DMA Channel by programming the SCFG_CHSEC0 register in the DMA Unit Security Configuration Register Frame.


Note

By default, all channels are set as Secure after reset.

If Security Extension is not present

If Security Extension is not present in your configuration, then the Security Configuration Register Frame is not present, and this step is skipped. All channels are fixed as Non-secure.

5.3 Step 3 Configuring the DMA Unit Level register

Secure privileged and Non-secure privileged software must configure the registers in the DMA Unit Secure Control Register Frame and the DMA Unit Non-secure Control Register Frame, respectively.


Note

The DMA Unit Secure Control Register Frame is only present if Security Extension is present in your configuration. If Security Extension is not present in your configuration, only the DMA Unit Non-secure Control registers must be programmed.

DMA Unit Secure Control Register Frame

Secure privileged software can:

- Configure each Secure channel as Privileged or Unprivileged.
- Set channel IDs if the CHID_WIDTH configuration parameter's value is greater than 0.
- Set Unit-level interrupts.

- Set the minimum power state.
- Enable CTI debugging.

Setting the privilege level and channel ID for each Secure channel

For setting the privilege levels and the Channel IDs:

1. Select the channel through the CHPTR register field of the SEC_CHPTR register.
2. Set the channel as Privileged or Unprivileged in the SEC_CHCFG.CHPRIV field.

   By default, all channels are set as Unprivileged after reset.
3. If the configuration uses CHID, program its value in the SEC_CHCFG.CHID field and set the SEC_CHCFG.CHIDVLD register field to 1.

Secure DMA Unit-level interrupts, minimum power state, and debug support

Secure Privileged software can program the SEC_CTRL register to enable DMA Unit-level interrupts, set the minimum power state and enable receiving halt and resume requests on the CTI interface for debugger support.

- If the DMA uses Secure Unit-level interrupts, enable the individual events in the SEC_CTRL register register.
- Program the minimum power state of the DMA Unit in the SEC_CTRL.DISMINPWR field. The DMA does not request lower power state than then programmed value, even it is inactive, i.e. all Secure channels are disabled or stopped.
- Enable idle channel retention by setting the SEC_CTRL.IDLERETEN register field if the DMA is allowed to go to retention when the Secure Channels that are enabled are waiting for an event, e.g. triggers.
- Secure software can enable CTI debugging in the SEC_CTRL.DBGHALTEN register field. Secure software can prevent Non-secure software from enabling CTI debugging by setting the DBGHALTNSRO register field to 1.

DMA Unit Non-secure Control Register Frame

Non-secure privileged software can

- Configure each Non-secure channel as Privileged or Unprivileged.
- Set Channel IDs if the CHID_WIDTH configuration parameter's value is greater than 0.
- Set Non-Secure Unit-level interrupts.
- Set the minimum power state.
- Enable CTI debugging, if permitted by Secure software.

Setting the privilege level and Channel ID for each Non-secure channel

For setting the privilege levels and the Channel IDs,

1. Select the Channel through the NSEC_CHPTR.CHPTR register field.
2. Set the channel as Privileged or Unprivileged in the NSEC_CHCFG.CHPRIV register field. Note: By default, all channels are set as Unprivileged after reset.
3. If the configuration uses CHID, program its value in the NSEC_CHCFG.CHID register field and set the NSEC_CHCFG.CHIDVLD register field to 1.

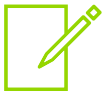
Non-secure DMA Unit-level interrupts, minimum power state, and debug support

Secure Privileged software can program the SEC_CTRL register to enable DMA Unit-level interrupts, set the minimum power state and enable halt and resume requests on the CTI interface for debugger support.

- If the DMA uses Non-secure Unit-level interrupts, enable the individual events in the NSEC_CTRL register.
- Program the minimum power state of the DMA Unit in the SEC_CTRL.DISMINPWR register field. The DMA does not request lower power state than then programmed value, even it is inactive, i.e. all Non-secure channels are disabled or stopped.
- Enable idle channel retention by setting the SEC_CTRL.IDLERETEN register field if the DMA is allowed to go to retention when the Non-secure Channels that are enabled are waiting for an event, e.g. triggers.
- Non-secure software can enable CTI debugging in the SEC_CTRL.DBGHALTEN register field. Secure software can prevent Non-secure software from enabling CTI debugging by setting the DBGHALTNSRO register field to 1.

5.4 Step 4 Configuring a DMA Channel command

To program a DMA Channel to perform a DMA command, the registers in the Channel's own DMA Channel Register Frame must be set up accordingly.

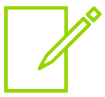


Channels set as Privileged can only be programmed by Privileged software.
Channels set as Secure can only be programmed by Secure software.

Channel Control Register

Configure the type of the transfers and the resources needed by the DMA command in the CH_CTRL register:

- Program the transfer entity size in the CH_CTRL.TRANSIZE register field.
- Program the Channel's priority in the CH_CTRL.CHPRIO register field.
- Set the Channel lock request in the CH_CTRL.CHLOCKRE Q register field if the virtual channel must keep its allocation.



If all physical channels are allocated to virtual channels that have their CH_CTRL.CHLOCKREQ bits sets, then the lock requests are ignored to avoid impeding forward progress for other non-allocated channels that are set as higher priority.

- Program the DMA operation type in the CH_CTRL.XTYPE register field.

- If the DMA command shall reload initial values at the end of a DMA command, program the automatic register reload type in the CH_CTRL.REGRELOADTYPE register field accordingly.
- Program when the STAT_DONE status flag is asserted during the command operation in the CH_CTRL.DONETYPE register field.
- If the DMA command shall automatically pause when the STAT_DONE flag is set, set the CH_CTRL.DONEPAUSEN register field.
- If your DMA configuration supports triggers and the current DMA Command uses triggers, program the CH_CTRL.USESRCTRIGIN, CH_CTRL.USEDESTRIGIN, and CH_CTRL.USETIGOUT register fields accordingly.
- If your DMA configuration supports GPOs and the current DMA Command uses GPOs, program the CH_CTRL.USEGPO register field.

Source and destination addresses

Program the source and destination starting addresses.

- Program the source address in the CH_SRCADDR register.
- Program the destination address in the CH_DESADDR register.

Number of DMA transfers

Program the number of transfers in your DMA Command in the CH_XSIZE.SRCXSIZE register field. CH_XSIZE.DESXSIZE automatically takes on the same value that you programmed into CH_XSIZE.SRCXSIZE.

Memory attributes

Program the source and destination memory attributes.

Program the source side memory attributes in the CH_SRCTRANSCFG register:

- Program the source memory type in the CH_SRCTRANSCFG.SRCMEMATTRLO and CH_SRCTRANSCFG.SRCMEMATTRHI register fields.
- Program the source transfer shareability attribute in the CH_SRCTRANSCFG.SRCSHAREATTR register field.
- Program the source transfer Non-secure attribute in the CH_SRCTRANSCFG.SRCNONSECATTR register field. When a channel is Non-secure this bit is tied to 1.
- Program the source transfer privilege attribute in the CH_SRCTRANSCFG.SRCPRIVATTR register field. When a channel is unprivileged this bit is tied to 0.
- Program the maximum allowed burst size for read transfers in the CH_SRCTRANSCFG.SRCMAXBURSTLEN register field.

Program the destination side memory attributes in the CH_DESTRANSCFG register:

- Program the destination memory type in the CH_DESTRANSCFG.DESMEMATTRLO and CH_DESTRANSCFG.DESMEMATTRHI register fields.
- Program the destination transfer shareability attribute in the CH_DESTRANSCFG.DESSHAREATTR register field.

- Program the destination transfer Non-secure attribute in the CH_DESTRANSCFG.DESNONSECATTR register field. When a channel is Non-secure this bit is tied to 1.
- Program the destination transfer privilege attribute in the CH_DESTRANSCFG.DESPRIVATTR register field. When a channel is unprivileged this bit is tied to 0.
- Program the maximum allowed burst size for write transfers in the CH_DESTRANSCFG.DESMAXBURSTLEN register field.

Address increments

- Program the source and destination address increments.
- Program the increment values used to update the source and destination addresses after each transferred data unit in the CH_XADDRINC register.
- Program the source address increment in the CH_XADDRIN.SRCXADDRINC register field.
- Program the destination address increment in the CH_XADDRIN.DESXADDRINC register field.

Triggers

If your DMA configuration supports triggers and the current DMA Command uses triggers, configure the source and destination triggers inputs and the trigger output.

Source trigger input

Set up the source trigger input in the CH_SRCTRIGINCFG register:

- Program the source trigger input type in the CH_SRCTRIGINCFG.SRCTRIGINTYPE register field depending on if the DMA command uses hardware or software triggers:
- Program the source trigger input mode in the CH_SRCTRIGINCFG.SRCTRIGINMODE register field depending on if the DMA command interprets the source side trigger input as command mode trigger, DMA-driven flow control trigger, or peripheral-driven flow control trigger.
- Program the source trigger block size in the CH_SRCTRIGINCFG.SRCTRIGINBLKSIZE register field.

Destination trigger input

Set up the destination trigger input in the CH_DESTRIGINCFG register:

- Program the destination trigger input type in the CH_SRCTRIGINCFG.DESTRIGINTYPE register field depending on if the DMA command uses hardware or software triggers.
- Program the destination trigger input mode in the CH_DESTRIGINCFG.DESTRIGINMODE register field depending on if the DMA command interprets the destination side trigger input as command mode trigger, DMA-driven flow control trigger, or peripheral-driven flow control trigger.
- Program the destination trigger block size in the CH_DESTRIGINCFG.DESTRIGINBLKSIZE register field.

Trigger output

Set up the trigger output in the CH_TRIGOUTCFG register:

- Program the trigger output type in the CH_TRIGOUTCFG.TRIGOUTTYPE register field depending on if the DMA command uses hardware or software triggers.

- Program the source trigger input mode in the CH_SRCTRIGINCFG.SRCTRIGINMODE register field depending on if the DMA command interprets the source side trigger input as command mode trigger, DMA-driven flow control trigger, or peripheral-driven flow control trigger.
- Program the source trigger block size in the CH_SRCTRIGINCFG.SRCTRIGINBLKSIZE register field.

GPOs

If your DMA configuration supports GPOs and the current DMA Command uses GPOs, configure their values to be used during the DMA command:

- Program the CH_GPOEN0 register to enable GPO bits.
- Program the GPO value in the CH_GPOVAL0 register.

Command link

If the DMA command uses command link, configure the address and the memory attributes for reading the command descriptor:

- Program the link address and enable command link in the CH_LINKADDR register's LINKADDR and LINKADDREN register fields, respectively.
- Program the memory attributes in the CH_LINKATTR register.
  - Program the command link memory type in the CH_LINKATTR.LINKMEMATTRLO and CH_LINKATTR.LINKMEMATTRHI register fields.
  - Program the command link transfer shareability attribute in the CH_LINKATTR.LINKSHAREATTR register field.


Note

Command link transfers use the DMA channel's security and privilege attributes.

Automatic restarting

If the DMA command uses automatic restart, configure the CH_AUTOCFG register.

- If the DMA command restarts a set number of times, program the number of automatic restarts in the CH_AUTOCF.CMDRESTARTCNT register field.
- If the DMA command restarts an indefinite number of times before software stops the restart loop by setting CH_CMD.DISABLECMD or CH_CMD.STOPCMD register field, set the CH_AUTOCF.CMDRESTARTINFEN register field to 1.


Note

The automatic restart functionality is closely related to the CH_CTRL.REGRELOADTYPE register field.

Channel Interrupts

If the DMA command uses interrupts, enable the individual events in the CH_INTREN register for which the DMA Channel shall generate an interrupt request.

Enable the command

After configuring the DMA command, enable it by setting the CH_CMD.ENABLECMD register field to 1.

5.5 Configuring burst generation on the Data AHB5 interface

The DMA channels can be configured to transfer several data elements from one location to another. There are multiple settings for every command that define how DMAC sends the transfers to the bus interface.

TRANSIZE

CH_CTRL.TRANSIZE setting adjusts the size of the data elements the command moves. The setting is the same for both read and write directions and ranging in power of 2 steps from a single byte to the number of bytes that fit into the DATA_WIDTH of the bus interface. This sets the base for all internal counters, the address increment logic, and the burst calculation logic.

XSIZE

The XSIZE setting defines the total number of transfers.

TRIGINBLKSIZE

The TRIGINBLKSIZE register field adjusts the maximum number of elements sent for a block trigger request. This can be used to split up the command into smaller parts. The total number of elements in the command is not necessarily a multiple of TRIGINBLKSIZE so partial blocks can also be sent at the end of a command. The TRIGINBLKSIZE setting can be different on source and destination sides which may result in having partial data stored temporarily in the FIFO until the triggers are acknowledged.

In the following example, the SRCTRIGINBLKSIZE is 5 and DESTRIGINBLKSIZE is 8, that results in having 2 triggers on the source side that generate enough data to the destination side but leaves 2 data elements in the FIFO until the next destination trigger is received.

1. Source trigger received – 5 elements read, FIFO level increased to 5.
2. Source trigger received – 5 elements read, FIFO level increased to 10.
3. Destination trigger received – 8 elements written, FIFO level decreased to 2.

Trigger events can happen in parallel and it depends on the FIFO size and its actual level whether the read or write side of the channel must wait for the other side to finish. Care must be taken when programming the trigger block size settings for timing critical applications.

MAXBURSTLEN

The MAXBURSTLEN register field limits the maximum number of beats sent out by the DMAC on the bus interface for a burst request. This setting can be used to adjust the bus utilization during the command and also creates arbitration points when multiple channels try to access the bus concurrently.

The following example shows how the XSIZE=17, TRIGINBLKSIZE=6 and MAXBURSTLEN=5 settings affect the command execution. Trigger blocks and bursts are only defined to have a maximum size so the DMA command can send smaller trigger blocks and bursts as seen in the following table.

1. The total number of transfers is defined by XSIZE=17.
2. The transfers are sent in blocks that is defined by TRIGINBLKSIZE=6 (6+6+5=17). There are 3 blocks in total.
3. Either AHB SINGLE or INCR4 transactions are sent as defined by MAXBURSTLEN=5.

   As the last block only contains 5 transfers, there are 2 transactions (INCR4 and SINGLE)

Based on these settings one example DMA command consists of XSIZE number of TRANSIZE-based data elements. This command can be split into several blocks defined by TRIGINBLKSIZE which can also be split into multiple bursts limited by the MAXBURSTLEN.

Table 5-1: Example of effects of different settings values

XSIZE = 17							
Block size = 6			Block size = 6			Block size = 5	
BURST length #1	BURST length #2	BURST length #3	BURST length #1	BURST length #2	BURST length #3	BURST length #1	BURST length #2
INCR4	SINGLE	SINGLE	INCR4	SINGLE	SINGLE	INCR4	SINGLE

The DMAC selects the largest burst possible to reduce the number of address requests on the bus, but this can be limited if the bandwidth or latency requirements require adjustments from software.

Burst generation is affected by the values of the DATA_WIDTH parameter, and the values of the TRANSIZE, XSIZE, TRIGINBLKSIZE and MAXBURSTLEN fields of Software programmable registers. In addition, burst generation is also affected by the following factors:

- The size of the FIFO limits the maximum amount of data in a burst.
- The DMAC only generates bursts that do not cross the 1kB address boundary.
- Certain transfer properties such as transfer type (device or memory) and aligned or unaligned starting and ending addresses can also affect how the DMAC generates bursts, as the DMAC tries to optimize bandwidth by utilizing as much of the bus width as possible and will group transfers if it can.

6. Programmers model

The Programmers model describes the memory maps and registers of the CoreLink DMA-250.

When using the programmers model, adhere to the following guidelines:

- 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.
  - Unless otherwise specified, all register bits are reset to a logic 0 by a system or power up reset.

The following describes the access type:

RW
: Read and write

RO
: Read-only

WO
: Write-only

RAZ
: Read-As-Zero

WI
: Writes ignored

W1C
: Write 1 to Clear

6.1 Memory map

The memory map includes the maximum number of implemented blocks.

Table 6-1: Memory map

Address range	Description
0x0000 - 0x00ff	DMA Unit Security Configuration Register Frame
0x0100 - 0x01ff	DMA Unit Secure Control Register Frame
0x0200 - 0x02ff	DMA Unit Non-secure Control Register Frame
0x0300 - 0x0eff	Reserved
0x0f00 - 0x0fff	DMA Unit Information Register Frame
0x1000 - 0x10ff	DMA Channel 0 Register Frame

Address range	Description
0x1100 - 0x11ff	DMA Channel 1 Register Frame
0x1200 - 0x12ff	DMA Channel 2 Register Frame
0x1300 - 0x13ff	DMA Channel 3 Register Frame
0x1400 - 0x14ff	DMA Channel 4 Register Frame
0x1500 - 0x15ff	DMA Channel 5 Register Frame
0x1600 - 0x16ff	DMA Channel 6 Register Frame
0x1700 - 0x17ff	DMA Channel 7 Register Frame
0x1800 - 0x18ff	DMA Channel 8 Register Frame
0x1900 - 0x19ff	DMA Channel 9 Register Frame
0x1a00 - 0x1aff	DMA Channel 10 Register Frame
0x1b00 - 0x1bff	DMA Channel 11 Register Frame
0x1c00 - 0x1cff	DMA Channel 12 Register Frame
0x1d00 - 0x1dff	DMA Channel 13 Register Frame
0x1e00 - 0x1eff	DMA Channel 14 Register Frame
0x1f00 - 0x1fff	DMA Channel 15 Register Frame

6.2 DMASECCFG summary, DMA Unit Security Configuration Register Frame

This block contains the configuration registers for the security related behavior of the DMAC. The security configuration is recommended to be defined before the operation of the DMAC and caution must be taken if the configuration is changed during runtime. This register frame can be accessed by Secure privileged accesses only.

Table 6-2: DMASECCFG register summary

Offset	Name	Type	Default	Width	Description
0x00	SCFG_CHSEC0	RW	0x0000_0000	32	-
0x08	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x28	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x40	SCFG_CTRL	RW	0x0000_0000	32	-
0x44	SCFG_INTRSTATUS	RW	0x0000_0000	32	-

DMASECCFG global constraints

The following global constraints apply to all registers in this block:

- Unprivileged accesses result in **RAZ/WI** response.
- Non-secure accesses result in **RAZ/WI** or error response depending on the security configuration of the DMAC.
- Base address: 0x0000
- Size: 0x0100

- Instances: 1

6.2.1 SCFG_CHSEC0

The Secure Configuration Channel Security Mapping register is used to define the security attribute of each channel from Channel 0 to NUM_CHANNELS - 1.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCFG

Offset

0x000

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after SEC_CFG_LCK is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-3: SCFG_CHSEC0 register bit descriptions

Bits	Name	Description	Type	Default
[31:(NUM_PCH * VCH_PCH_RATIO)]	-	Reserved	-	-
[(NUM_PCH * VCH_PCH_RATIO)-1:0]	SCFGCHSEC0	Secure Configuration Channel Security Mapping for channel 0 to 31. When [i] set to '1', CH[i] is Non-secure world, else Secure world. The (NUM_PCH * VCH_PCH_RATIO) parameter limits this field, unused bits are **RAZ/WI**. Value for bit[i] can only be changed if the selected channel is not enabled, so reading back the register content is needed to check the success of the change. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0

6.2.2 SCFG_CTRL

The Secure Configuration Control register defines the behavior of the DMAC when security violation occurs and also allows the security configuration to be locked until the next reset.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCFG

Offset

0x040

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after SEC_CFG_LCK is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-4: SCFG_CTRL register bit descriptions

Bits	Name	Description	Type	Default
[31]	SEC_CFG_LCK	Security Configuration Lock. When set to '1', only SCFG.STAT_SECACCVIO can be cleared, all other register fields in the this block become read-only. Once set to '1', this field can only be set back to '0' by reset. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[30:2]	Reserved	-	**RAZ/WI**	-
[1]	RSPTYPE_SECACCVIO	Secure Access Violation response type configuration. • 0: RAZ/WI • 1: bus error The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT.	RW	0x0
[0]	INTREN_SECACCVIO	Secure Access Violation Interrupt Enable. • 0: no interrupt • 1: interrupt raised for security violation The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT.	RW	0x0

6.2.3 SCFG_INTRSTATUS

The Secure Configuration Interrupt Status register shows the interrupt status for security access violations.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCFG

Offset

0x044

Type

RW

Default

0x000_00000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-5: SCFG_INTRSTATUS register bit descriptions

Bits	Name	Description	Type	Default
[31:17]	Reserved	-	**RAZ/WI**	-
[16]	STAT_SECACCVIO	Secure Access Violation Status. Set to '1' when a security violation occurred. Write '1' to clear. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1C	0x0
[15:1]	Reserved	-	**RAZ/WI**	-
[0]	INTR_SECACCVIO	Secure Access Violation Interrupt Status. Set to '1' when SCFG.STAT_SECACCVIO is asserted and SCFG.INTREN_SECACCVIO = 1. Cleared automatically when STAT_SECACCVIO is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RO	0x0

6.3 DMASECCTRL summary, DMA Unit Secure Control Register Frame

This block contains the configuration registers for all Secure channels and other Secure resources of the DMAC. The Secure software can control and monitor the status of the channels before and

during the execution of the DMA commands. Some of the registers become read-only when the ENABLECMD bit is set for the selected channel. This register frame can be accessed by Secure privileged accesses only.


Note

Certain register fields are stored in each VCH instance, others are stored in each PCH instance and a shadow copy is stored in the Context memory.

Table 6-6: DMASECCTRL register summary

Offset	Name	Type	Default	Width	Description
0x00	SEC_CHINTRSTATUS0	RO	0x0000_0000	32	-
0x08	SEC_STATUS	RW	0x0000_0000	32	-
0x0C	SEC_CTRL	RW	0x0000_0000	32	-
0x10	SEC_CNTXBASE	RW	0x0000_0000	32	-
0x14	SEC_CHPTR	RW	0x0000_0000	32	-
0x18	SEC_CHCFG	RW	0x0000_0000	32	SEC_CHCFG.CHIDVLD is stored in each VVCH instance SEC_CHCFG.CHID is stored in each PCH instance
0xF0	SEC_STATUSPTR	RW	0x0000_0000	32	-
0xF4	SEC_STATUSVAL	RO	0x0000_0000	32	-
0xF8	SEC_SIGNALPTR	RW	0x0000_0000	32	-
0xFC	SEC_SIGNALVAL	RW	0x0000_0000	32	-

DMASECCTRL global constraints

The following global constraints apply to all registers in this block:

- Unprivileged accesses result in **RAZ/WI** response.
- Non-secure accesses result in **RAZ/WI** or error response depending on the security configuration of the DMAC.
- Base address: 0x0100
- Size: 0x0100
- Instances: 1

6.3.1 SEC_CHINTRSTATUS0

The Secure Channel Interrupt Status register 0 shows the overall interrupt status of every Secure channel from Channel 0 to NUM_CHANNELS - 1.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x000

Type

RO

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-7: SEC_CHINTRSTATUS0 register bit descriptions

Bits	Name	Description	Type	Default
[31:(NUM_PCH * VCH_PCH_RATIO)]	Reserved	-	**RAZ/WI**	-
[(NUM_PCH * VCH_PCH_RATIO)-1:0]	CHINTRSTATUS0	Collated Non-secure Channel Interrupt flags for channel 0 to (NUM_PCH * VCH_PCH_RATIO) - 1. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RO	0x0

6.3.2 SEC_STATUS

The Secure Status register provides information about the overall status of the Secure channels.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x008

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-8: SEC_STATUS register bit descriptions

Bits	Name	Description	Type	Default
[31:21]	Reserved	-	**RAZ/WI**	-
[20]	STAT_CTXERR	Virtual Channel Context Memory Access Error Status. Set to '1' whenever accessing the virtual channel context memory results in an error. All secure channels get a stop request while this bit is set even if they were enabled after the error event. Cleared by writing '1' to this bit. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1C	0x0
[19]	STAT_ALLCHPAUSED	All Secure Channel Paused Status. Set to '1' whenever all Secure DMA channel is in paused or inactive state and the ALLCHPAUSE request is active. Not set when channels reach the paused state for other reasons. All Secure channels are forced to an immediate pause until ALLCHPAUSE request is asserted even if they were enabled after the request. Cleared by writing '1' to this bit which results in all Secure channels to resume their operation. The ALLCHPAUSE request is also cleared when this bit is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1C	0x0
[18]	STAT_ALLCHSTOPPED	All Secure Channel Stopped Status. Set to '1' whenever all Secure DMA channels are in stopped or inactive state and the ALLCHSTOP request is active. Not set when channels reach the stopped state for other reasons. All Secure channels are forced to an immediate stopped state until ALLCHSTOP request is asserted even if they were enabled after the request. Cleared by writing '1' to this bit. The ALLCHSTOP request is also cleared when this bit is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1C	0x0
[17]	STAT_ALLCHIDLE	All Secure Channel Idle Status. Set to '1' whenever all Secure DMA channel is in idle state after at least one channel was running. Cleared by writing '1' to this bit. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1C	0x0
[16:4]	Reserved	-	**RAZ/WI**	-
[3]	INTR_ALLCHPAUSED	All Secure Channel Paused Interrupt Status. Set to '1' when SECCTRL.STAT_ALLCHPAUSED is asserted and SECCTRL.INTREN_ALLCHPAUSED = 1. Cleared automatically when STAT_ALLCHPAUSED is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RO	0x0
[2]	INTR_ALLCHSTOPPED	All Secure Channel Stopped Interrupt Status. Set to '1' when SECCTRL.STAT_ALLCHSTOPPED is asserted and SECCTRL.INTREN_ALLCHSTOPPED = 1. Cleared automatically when STAT_ALLCHSTOPPED is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RO	0x0
[1]	INTR_ALLCHIDLE	All Secure Channel Idle Interrupt Status. Set to '1' when SECCTRL.STAT_ALLCHIDLE is asserted and SECCTRL.INTREN_ALLCHIDLE = 1. Cleared automatically when STAT_ALLCHIDLE is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RO	0x0
[0]	INTR_ANYCHINTR	Combined Secure Channel Interrupt Flag. Set to '1' when any Secure channel has an interrupt request in CHINTRSTATUS0 and SECCTRL.INTREN_ANYCHINTR = 1. Cleared automatically when the source of the channel interrupt is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RO	0x0

6.3.3 SEC_CTRL

The Secure Control registers can be used to adjust the interrupt generation and general behavior of the Secure channels.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x00C

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-9: SEC_CTRL register bit descriptions

Bits	Name	Description	Type	Default
[31:30]	DISMINPWR	Minimum Power state of the DMAC when at least one Secure channel is present. • 00: OFF • 01: Retention • 10: ON • Others: Reserved The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[29]	IDLERETEN	Idle Channel Retention Enable. Allows retention for Secure channels that are enabled and waiting for an event in IDLE state. • 0: disabled • 1: enabled The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT & (1)	RW	0x0

Bits	Name	Description	Type	Default
[28]	DBGHALTEN	Debug Halt Enabled. When set to '0', the DMA ignores the halt request from an external debugger. Clearing this bit while halt is ongoing results in continuing the operation. When set to '1', the debugger request to halt the DMA is allowed for all channels. This field is common for Non-secure and Secure side of the DMA, but control can be limited by the SEC_CTRL.DBGHALTNSRO register field. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[27]	DBGHALTNSRO	Debug Halt Enable Non-secure Read Only. When set to '1', the NSEC_CTRL.DBGHALTEN register becomes read-only. When set to '0', NSEC_CTRL.DBGHALTEN has read-write access. This register allows the Secure software to limit the Non-secure sofware adjusting the common DBGHALTEN register bit. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[26:10]	Reserved	-	**RAZ/WI**	-
[9]	ALLCHPAUSE	Secure All Channel Pause Request. When set to '1', all Secure channels get a pause request. Stays asserted until the channels are paused and the STAT_ALLCHPAUSED status flag is set. Cleared automatically when the STAT_ALLCHPAUSED status flag is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1S	0x0
[8]	ALLCHSTOP	Secure All Channel Stop Request. When set to '1', all Secure channels get a stop request. Stays asserted until the STAT_ALLCHSTOPPED status flag is set. Cleared automatically when the STAT_ALLCHSTOPPED status flag is cleared. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1S	0x0
[7:5]	Reserved	-	**RAZ/WI**	-
[4]	INTREN_CTXERR	Virtual Channel Context Memory Access Error Interrupt Enable | RW | 0x0 |		
[3]	INTREN_ALLCHPAUSED	All Secure Channel Paused Interrupt Enable The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[2]	INTREN_ALLCHSTOPPED	All Secure Channels Stopped Interrupt Enable The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[1]	INTREN_ALLCHIDLE	All Secure Channel Idle Interrupt Enable The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[0]	INTREN_ANYCHINTR	Combined Secure Channel Interrupt Enable The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0

6.3.4 SEC_CNTXBASE

Secure Channel Context Area Base Pointer

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x010

Type

RW

Default

0x00000000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-10: SEC_CNTXBASE register bit assignments

Bits	Name	Description	Type	Default
[31:0]	CNTXBASE	Secure Channel Context Area Base Pointer	RW	0x0

6.3.5 SEC_CHPTR

The Secure Channel Pointer register is used to select one channel that needs to be configured through the following registers that have SEC_CH prefix.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x014

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-11: SEC_CHPTR register bit descriptions

Bits	Name	Description	Type	Default
[31:6]	Reserved	-	**RAZ/WI**	-
[5:0]	CHPTR	Secure Channel Pointer. Selects which channel settings can be adjusted by the following registers. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0

6.3.6 SEC_CHCFG

The Secure Channel Configuration register provides configuration fields for channel attributes.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x018

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set for the channel selected by CHPTR.

There are these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-12: SEC_CHCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:18]	Reserved	-	**RAZ/WI**	-
[17]	CHPRIV	Secure Channel Privilege Enable. SECCHPTR selects the channel. When set to '1' it allows the channel to send transfers marked as Privileged only. The configuration registers of the selected channel are also given privileged only access rights. When set to '0' the channel is only allowed to send unprivileged transfers and the channel registers can be accessed by both privileged and unprivileged register accesses. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0
[16]	CHIDVLD	Secure Channel ID valid. SECCHPTR selects the channel. Set to '1' to drive the channel ID value in CHID for all the transfers by the selected channel. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT & (CHID_WIDTH > 0)	RW	0x0

Bits	Name	Description	Type	Default
[15:0]	CHID	Secure Channel ID value. SECCHPTR selects the channel ID value to be read or written via this register. CHID_WIDTH limits this field, unused bits are **RAZ/WI**. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT & (CHID_WIDTH > 0)	RW	0x0

6.3.7 SEC_STATUSPTR

The Secure Unit Status Pointer register can set a pointer to an internal status register that can show the global state of the Secure channels.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x0F0

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-13: SEC_STATUSPTR register bit descriptions

Bits	Name	Description	Type	Default
[31:4]	Reserved	-	**RAZ/WI**	-

Bits	Name	Description	Type	Default
[3:0]	SECSTATUSPTR	Secure DMA Unit status pointer used to select which status value to view using STATUSVALUE register. Pointer values are: • 0: Channel Enabled Status for channel numbers [31:0] • 1: Reserved • 2: Channel Stopped Status for channel numbers [31:0] • 3: Reserved • 4: Channel Paused Status for channel numbers [31:0]. • Others: Reserved. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0

6.3.8 SEC_STATUSVAL

The Secure Unit Status Value register shows the current value of a status register on Secure channels selected by the pointer in SEC_STATUSPTR.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x0F4

Type

RO

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-14: SEC_STATUSVAL register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	SECSTATUSVAL	Secure DMA Unit status value. Can be used for reading internal status values of the DMA Unit for debug purposes. Values shown here are dependent on STATUSPTR. Note that inputs are masked by security mapping before being presented here. This means that only status values mapped to Secure world are visible as non-Zero values. All unimplemented bits are **RAZ/WI**. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RO	`0x0`

6.3.9 SEC_SIGNALPTR

The Secure Unit Signal Pointer register can set a pointer to an internal register that shows the state of the Secure interfaces attached to the DMAC.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

`0x0F8`

Type

RW

Default

`0x0000_0000`

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-15: SEC_SIGNALPTR register bit descriptions

Bits	Name	Description	Type	Default
[31:4]	Reserved	-	**RAZ/WI**	-

Bits	Name	Description	Type	Default
[3:0]	SECSIGNALPTR	Secure DMA Unit signal pointer used to select which inputs to view using SIGNALVAL register. Pointer values, x, are: • 0 to 7: Trigger Input Requests [(31+32*x*):*32*x] • 8 to 9: Trigger output Acknowledges [(31+32(*x-8*)):*32*(x-8)] • 10 to 11: GPO output value [(31+32(*x-10*)):(*32*(x-10))], only present when HAS_GPOSEL is set. • Others: Reserved The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	RW	0x0

6.3.10 SEC_SIGNALVAL

The Secure Unit Signal Value register shows the current value of the Secure interfaces selected by the pointer in SEC_SIGNALPTR.

Configurations

See bit descriptions.

Attributes

Register frame

DMASECCTRL

Offset

0x0FC

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

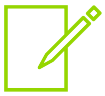
The following table shows the register bit assignments.

Table 6-16: SEC_SIGNALVAL register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	SECSIGNALVAL	Secure DMA Unit signal status. Can be used for reading signal values of triggers and GPOs for debug. Values shown here are dependent on SIGNALPTR. Note that inputs are masked by security mapping before being presented here. This means that only signal values mapped to Secure world are visible as non-zero values. Writing to this register has the following effect: • Writing '1' to a Trigger Input Request that are not selected by any DMA channel causes a deny response for it. This can attempt to clear an unwanted trigger input. • Writing '1' to any Trigger Output Acknowledges status, GPO value and all Trigger Input Request selected by any DMA channel is ignored. All unimplemented bits are **RAZ/WI**. The field is **RAZ/WI** when the following condition is False: SECEXT_PRESENT	W1C	0x0

6.4 DMANSECCTRL summary, DMA Unit Non-secure Control Register Frame

This block contains the configuration registers for all Non-secure channels and other Non-secure resources of the DMAC. The Non-secure software can control and monitor the status of the channels before and during the execution of the DMA commands. Some of the registers become read-only when the ENABLECMD bit is set for the selected channel. This register frame can be accessed by both secure privileged and Non-secure privileged accesses.


Note

Certain register fields are stored in each VCH instance, others are stored in each PCH instance and a shadow copy is stored in the Context Memory (CTXMEM).

Table 6-17: DMANSECCTRL register summary

Offset	Name	Type	Default	Width	Description
0x00	NSEC_CHINTRSTATUS0	RO	0x0000_0000	32	-
0x08	NSEC_STATUS	RW	0x0000_0000	32	-
0x0C	NSEC_CTRL	RW	0x0000_0000	32	-
0x10	NSEC_CNTXBASE	RW	0x0000_0000	32	-
0x14	NSEC_CHPTR	RW	0x0000_0000	32	-
0x18	NSEC_CHCFG	RW	0x0000_0000	32	NSEC_CHCFG.CHIDVLD is stored in each VCH instance NSEC_CHCFG.CHID is stored in each PCH instance
0xF0	NSEC_STATUSPTR	RW	0x0000_0000	32	-
0xF4	NSEC_STATUSVAL	RO	0x0000_0000	32	-
0xF8	NSEC_SIGNALPTR	RW	0x0000_0000	32	-

Offset	Name	Type	Default	Width	Description
0xFC	NSEC_SIGNALVAL	RW	0x0000_0000	32	-

DMANSECCTRL global constraints

The following global constraints apply to all registers in this block:

- Unprivileged accesses result in **RAZ/WI** response.
- Base address: 0x0200
- Size: 0x0100
- Instances: 1

6.4.1 NSEC_CHINTRSTATUS0

The Non-Secure Channel Interrupt Status register 0 shows the overall interrupt status of every Non-secure channel from Channel 0 to NUM_CHANNELS - 1.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x000

Type

RO

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-18: NSEC_CHINTRSTATUS0 register bit descriptions

Bits	Name	Description	Type	Default
[31:(NUM_PCH * VCH_PCH_RATIO)]	Reserved	-	RAZ/WI	-
[(NUM_PCH * VCH_PCH_RATIO)-1:0]	CHINTRSTATUS0	Collated Non-secure Channel Interrupt flags for channel 0 to (NUM_PCH * VCH_PCH_RATIO) - 1.	RO	0x0

6.4.2 NSEC_STATUS

The Non-secure Status register provides information about the overall status of the Non-secure channels.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x008

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-19: NSEC_STATUS register bit descriptions

Bits	Name	Description	Type	Default
[31:20]	Reserved	-	RAZ/WI	-
[19]	STAT_ALLCHPAUSED	All Non-secure Channel Paused Status. Set to '1' whenever all Non-secure DMA channel is in paused or inactive state and the ALLCHPAUSE request is active. Not set when channels reach the paused state for other reasons. All Non-secure channels are forced to an immediate pause until ALLCHPAUSE request is asserted even if they were enabled after the request. Cleared by writing '1' to this bit which results in all Non-secure channels to resume their operation. The ALLCHPAUSE request is also cleared when this bit is cleared.	W1C	0x0
[18]	STAT_ALLCHSTOPPED	All Non-secure Channel Stopped Status. Set to '1' whenever all Non-secure DMA channels are in stopped or inactive state and the ALLCHSTOP request is active. Not set when channels reach the stopped state for other reasons. All Non-secure channels are forced to an immediate stopped state until ALLCHSTOP request is asserted even if they were enabled after the request. Cleared by writing '1' to this bit. The ALLCHSTOP request is also cleared when this bit is cleared.	W1C	0x0
[17]	STAT_ALLCHIDLE	All Non-secure Channel Idle Status. Set to '1' whenever all Non-secure DMA channel is in idle state after at least one channel was running. Cleared by writing '1' to this bit.	W1C	0x0
[16:4]	Reserved	-	RAZ/WI	-

Bits	Name	Description	Type	Default
[3]	INTR_ALLCHPAUSED	All Non-secure Channel Paused Interrupt Status. Set to '1' when NSECCTRL.STAT_ALLCHPAUSED is asserted and NSECCTRL.INTREN_ALLCHPAUSED = 1. Cleared automatically when STAT_ALLCHPAUSED is cleared.	RO	0x0
[2]	INTR_ALLCHSTOPPED	All Non-secure Channel Stopped Interrupt Status. Set to '1' when NSECCTRL.STAT_ALLCHSTOPPED is asserted and NSECCTRL.INTREN_ALLCHSTOPPED = 1. Cleared automatically when STAT_ALLCHSTOPPED is cleared.	RO	0x0
[1]	INTR_ALLCHIDLE	All Non-secure Channel Idle Interrupt Status. Set to '1' when NSECCTRL.STAT_ALLCHIDLE is asserted and NSECCTRL.INTREN_ALLCHIDLE = 1. Cleared automatically when STAT_ALLCHIDLE is cleared.	RO	0x0
[0]	INTR_ANYCHINTR	Combined Non-secure Channel Interrupt Flag. Set to '1' when any Non-secure channel has an interrupt request in CHINTRSTATUS0 and NSECCTRL.INTREN_ANYCHINTR = 1. Cleared automatically when the source of the channel interrupt is cleared.	RO	0x0

6.4.3 NSEC_CTRL

The Non-secure Control registers can be used to adjust the interrupt generation and general behavior of the Non-secure channels.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x00C

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-20: NSEC_CTRL register bit descriptions

Bits	Name	Description	Type	Default
[31:30]	DISMINPWR	Minimum Power state of the DMAC when at least one Non-secure channel is present. • 00: OFF • 01: Retention • 10: ON • Others: Reserved	RW	0x0
[29]	IDLERETEN	Idle Channel Retention Enable. Allows retention for Non-secure channels that are enabled and waiting for an event in IDLE state. • 0: disabled • 1: enabled	RW	0x0
[28]	DBGHALTEN	Debug Halt Enabled. • 0: the DMA ignores the halt request from an external debugger. Clearing this bit while halt is ongoing results in continuing the operation. • 1: the debugger request to halt the DMA is allowed. This field is common for Non-secure and Secure side of the DMA, but access to this register is limited by the DBGHALTNSRO register field.	RW	0x0
[27]	DBGHALTNSRO	Debug Halt Enable Non-secure Read Only. • 0: NSEC_CTRL.DBGHALTEN has read-write access. • 1: The NSEC_CTRL.DBGHALTEN register becomes read-only. This register is read-only for the Non-secure software.	RO	0x0
[26:10]	Reserved	-	**RAZ/WI**	-
[9]	ALLCHPAUSE	Non-secure All Channel Pause Request. When set to '1', all Non-secure channels get a pause request. Stays asserted until the channels are paused and the STAT_ALLCHPAUSED status flag is set. Cleared automatically when the STAT_ALLCHPAUSED status flag is cleared.	W1S	0x0
[8]	ALLCHSTOP	Non-secure All Channel Stop Request. When set to '1', all Non-secure channels get a stop request. Stays asserted until the STAT_ALLCHSTOPPED status flag is set. Cleared automatically when the STAT_ALLCHSTOPPED status flag is cleared.	W1S	0x0
[7:5]	Reserved	-	**RAZ/WI**	-
[4]	INTREN_CTXERR	Virtual Channel Context Memory Access Error Interrupt Enable	RW	0x0
[3]	INTREN_ALLCHPAUSED	All Non-secure Channel Paused Interrupt Enable	RW	0x0
[2]	INTREN_ALLCHSTOPPED	All Non-secure Channels Stopped Interrupt Enable	RW	0x0
[1]	INTREN_ALLCHIDLE	All Non-secure Channel Idle Interrupt Enable	RW	0x0
[0]	INTREN_ANYCHINTR	Combined Non-secure Channel Interrupt Enable	RW	0x0

6.4.4 NSEC_CNTXBASE

Non-Secure Channel Context Area Base Pointer

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x010

Type

RW

Default

0x00000000

Usage constraints

There are no usage constraints apart from these global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-21: NSEC_CNTXBASE register bit assignments

Bits	Name	Description	Type	Default
[31:0]	CNTXBASE	Non-Secure Channel Context Area Base Pointer	RW	0x0

6.4.5 NSEC_CHPTR

The Non-secure Channel Pointer register is used to select one channel that needs to be configured through the following registers through the NSEC_CHCFG.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x014

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from Global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-22: NSEC_CHPTR register bit descriptions

Bits	Name	Description	Type	Default
[31:6]	Reserved	-	RAZ/WI	-
[5:0]	CHPTR	Non-secure Channel Pointer. Selects which channel settings can be adjusted by the following registers.	RW	0x0

6.4.6 NSEC_CHCFG

The Non-secure Channel Configuration register provides configuration fields for channel attributes for the channel selected by NSEC_CHPTR.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x018

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set for the channel selected by CHPTR.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-23: NSEC_CHCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:18]	Reserved	-	**RAZ/WI**	-
[17]	CHPRIV	Non-secure Channel Privilege Enable. NSEC_CHPTR selects the channel. When set to '1' it allows the channel to send transfers marked as Privileged only. The configuration registers of the selected channel are also given privileged only access rights. When set to '0' the channel is only allowed to send unprivileged transfers and the channel registers can be accessed by both privileged and unprivileged register accesses.	RW	0x0
[16]	CHIDVLD	Non-secure Channel ID valid. NSEC_CHPTR selects the channel. Set to '1' to drive the channel ID value in CHID for all the transfers by the selected channel. The field is **RAZ/WI** when the following condition is False: CHID_WIDTH > 0	RW	0x0
[15:0]	CHID	Non-secure Channel ID value. NSEC_CHPTR selects the channel ID value to be read or written via this register. CHID_WIDTH limits this field, unused bits are **RAZ/WI**. The field is **RAZ/WI** when the following condition is False: CHID_WIDTH > 0	RW	0x0

6.4.7 NSEC_STATUSPTR

The Non-secure Unit Status Pointer register can set a pointer to an internal status register that can show the global state of the Non-secure channels.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x0F0

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-24: NSEC_STATUSPTR register bit descriptions

Bits	Name	Description	Type	Default
[31:4]	Reserved	-	**RAZ/WI**	-

Bits	Name	Description	Type	Default
[3:0]	NSECSTATUSPTR	Non-secure DMA Unit status pointer used to select which status value to view using STATUSVALUE register. Pointer values are: • 0: Channel Enabled Status for channel numbers [31:0]. • 1: Reserved • 2: Channel Stopped Status for channel numbers [31:0]. • 3: Reserved • 4: Channel Paused Status for channel numbers [31:0]. • Others: Reserved.	RW	0x0

6.4.8 NSEC_STATUSVAL

The Non-secure Unit Status Value register shows the current value of a status register on Non-secure channels selected by the pointer in NSEC_STATUSPTR.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

0x0F4

Type

RO

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-25: NSEC_STATUSVAL register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	NSECSTATUSVAL	Non-secure DMA Unit status value. Can be used for reading internal status values of the DMA Unit for debug purposes. Values shown here are dependent on STATUSPTR. Note that inputs are masked by security mapping before being presented here. This means that only status values mapped to Non-secure world are visible as non-Zero values. All unimplemented bits are **RAZWI**.	RO	0x0

6.4.9 NSEC_SIGNALPTR

The Non-secure Unit Signal Pointer register can set a pointer to an internal register that shows the state of the Non-secure interfaces attached to the DMAC.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

`0x0F8`

Type

RW

Default

`0x0000_0000`

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-26: NSEC_SIGNALPTR register bit descriptions

Bits	Name	Description	Type	Default
[31:4]	Reserved	-	RAZ/WI	-
[3:0]	NSECSIGNALPTR	Non-secure DMA Unit signal pointer used to select which inputs or outputs to view using SIGNALVAL register. Pointer values, x, are: • 0: Source Trigger Input Requests [31:0]. Limited by (NUM_PCH * VCH_PCH_RATIO), all unimplemented bits are reserved. • 1: Destination Trigger Input Requests [31:0]. Limited by (NUM_PCH * VCH_PCH_RATIO), all unimplemented bits are reserved. • 8: Trigger output Acknowledges [31:0] • 10 to 11: GPO output value [(31+32*(x-10)):(32*(x-10))], only present when HAS_GPOSEL is set • Others: Reserved	RW	`0x0`

6.4.10 NSEC_SIGNALVAL

The Non-secure Unit Signal Value register shows the current value of the Non-secure interfaces selected by the pointer in NSEC_SIGNALPTR.

Configurations

See bit descriptions.

Attributes

Register frame

DMANSECCTRL

Offset

`0x0FC`

Type

RW

Default

`0x0000_0000`

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-27: NSEC_SIGNALVAL register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	NSECSIGNALVAL	Non-secure DMA Unit signal status. Can be used for reading signal values of triggers and GPOs for debug. Values shown here are dependent on SIGNALPTR. Note that inputs are masked by security mapping before being presented here. This means that only signals mapped to Non-secure world are visible as non-zero values. Writing to this register has the following effect: • Writing '1' to a Non-secure Trigger Input Request that are not selected by any DMA channel causes a deny response for it. This can attempt to clear an unwanted trigger input. • Writing '1' to any Trigger Output Acknowledges status, GPO status and all Trigger Input Request already selected by any DMA channel is ignored. All unimplemented bits are **RAZWI**.	W1C	`0x0`

6.5 DMAINFO summary, DMA Unit Information Register Frame

This frame provides information about the capabilities and parameters of the DMA unit. The registers in this frame can be accessed by all types of accesses.

- Base address: `0x0F00`
- Size: `0x0100`
- Instances: 1

Table 6-28: DMAINFO register summary

Offset	Name	Type	Default	Width	Description
`0xB0`	DMA_BUILDCFG0	RO	IMPL_DEF	32	-
`0xB4`	DMA_BUILDCFG1	RO	IMPL_DEF	32	-
`0xB8`	DMA_BUILDCFG2	RO	IMPL_DEF	32	-
`0xC8`	IIDR	RO	IMPL_DEF	32	-
`0xCC`	AIDR	RO	IMPL_DEF	32	-
`0xD0`	PIDR4	RO	IMPL_DEF	32	-
`0xE0`	PIDR0	RO	IMPL_DEF	32	-
`0xE4`	PIDR1	RO	IMPL_DEF	32	-
`0xE8`	PIDR2	RO	IMPL_DEF	32	-
`0xEC`	PIDR3	RO	`0x0000_0000`	32	-
`0xF0`	CIDR0	RO	`0x0000_000D`	32	-
`0xF4`	CIDR1	RO	`0x0000_00F0`	32	-
`0xF8`	CIDR2	RO	`0x0000_0005`	32	-
`0xFC`	CIDR3	RO	`0x0000_00B1`	32	-

6.5.1 DMA_BUILDCFG0

The DMA Build Configuration register 0 is used to provide information about the configuration parameters of the DMAC.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0B0`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-29: DMA_BUILDCFG0 register bit descriptions

Bits	Name	Description	Type	Default
[31:25]	Reserved	-	**RAZ/WI**	-
[24:20]	CHID_WIDTH	Channel ID Width. '0' means CHID is not present.	RO	CHID_WIDTH
[19]	Reserved	-	**RAZ/WI**	-
[18:16]	DATA_WIDTH	Data Width. • 000: 8-bit • 001: 16-bit • 010: 32-bit • 011: 64-bit • 100: 128-bit • 101: 256-bit • 110: 512-bit • 111: 1024-bit	RO	log2(DATA_WIDTH / 8)
[15:10]	ADDR_WIDTH	Address Width in bits = ADDR_WIDTH + 1	RO	ADDR_WIDTH - 1
[9:4]	NUM_CHANNELS	Number of Channels + 1 supported by the DMAC	RO	(NUM_PCH * VCH_PCH_RATIO) - 1
[3]	Reserved	-	**RAZ/WI**	-
[2:0]	FRAMETYPE	Register Frame Type. • 000: Combined Frame • 001: Security Configuration Frame • 010: Secure Control Frame, or if TrustZone not implemented, Control Frame. • 011: Non-secure Control Frame • 100: DMA Channel Frame. Note that each frame replicates the Information fields.	RO	`0x0`

6.5.2 DMA_BUILDCFG1

The DMA Build Configuration register 1 is used to provide information about the configuration parameters of the DMAC.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0B4`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-30: DMA_BUILDCFG1 register bit descriptions

Bits	Name	Description	Type	Default
[31:25]	Reserved	-	**RAZ/WI**	-
[24]	Reserved	-	**RAZ/WI**	-
[23:17]	Reserved	-	**RAZ/WI**	-
[16]	HAS_TRIGSEL	Has Selectable Trigger Support	RO	`0x0`
[15:9]	NUM_TRIGGER_OUT	Number of Triggers Outputs. '0' means that no output triggers are present. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	RO	(HAS_TRIG * NUM_PCH * VCH_PCH_RATIO)
[8:0]	NUM_TRIGGER_IN	Number of Triggers Inputs. '0' means that no input triggers are present. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	RO	(HAS_TRIG * NUM_PCH * VCH_PCH_RATIO * 2)

6.5.3 DMA_BUILDCFG2

The DMA Build Configuration register 2 is used to provide information about the configuration parameters of the DMAC.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0B8`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-31: DMA_BUILDCFG2 register bit descriptions

Bits	Name	Description	Type	Default
[31:12]	Reserved	-	**RAZ/WI**	-
[11]	Reserved	-	**RAZ/WI**	-
[10]	HAS_VCH	Has allocable DMA Channel Support.	**RAZ/WI**	`0x1`
[9]	HAS_RET	Has Retention Support	RO	`0x1`
[8]	HAS_TZ	Has TrustZone Support	RO	SECEXT_PRESENT
[7]	HAS_GPOSEL	Has Shared and hence Selectable GPO Support.	RO	`0x0`
[6:0]	Reserved	-	**RAZ/WI**	-

6.5.4 IIDR

The Implementation Identification register provides information about the implementer of the DMAC.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

0x0C8

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-32: IIDR register bit descriptions

Bits	Name	Description	Type	Default
[31:20]	PRODUCTID	Indicates the product ID	RO	0x250
[19:16]	VARIANT	Indicates the major revision, or variant, of the product rxpy identifier	RO	0x0
[15:12]	REVISION	Indicates the minor revision of the product rxpy identifier	RO	0x0
[11:0]	IMPLEMENTER	Contains the JEP106 code of the company that implemented the IP: • [11:8] - JEP106 continuation code of implementer • [7] - Always 0 • [6:0] - JEP106 identity code of implementer For Arm this field reads as 0x43B.	RO	0x43b

6.5.5 AIDR

The Architecture Identification register provides information about the architecture revision supported by the DMAC.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

0x0CC

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-33: AIDR register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	**RAZ/WI**	-
[7:4]	ARCH_MAJOR_REV	Architecture Major Revision.	RO	`0x0`
[3:0]	ARCH_MINOR_REV	Architecture Minor Revision.	RO	`0x1`

6.5.6 PIDR4

The Peripheral ID4 register returns byte[4] of the peripheral ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0D0`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-34: PIDR4 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	RAZ/WI	-
[7:4]	SIZE	4KB Count - the number of 4K pages used.< • 0x00: 4K • 0x01: 8K • 0x02: 16K • 0x03: 32K	RO	ceil(NUM_PCH * VCH_PCH_RATIO / 16)
[3:0]	DES_2	JEP106 Continuation Code	RO	0x4

6.5.7 PIDR0

The Peripheral ID0 register returns byte[0] of the peripheral ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

0x0E0

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-35: PIDR0 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	RAZ/WI	-
[7:0]	PART_0	Part Number [7:0]	RO	0x50

6.5.8 PIDR1

The Peripheral ID1 register returns byte[1] of the peripheral ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0E4`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-36: PIDR1 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	**RAZ/WI**	-
[7:4]	DES_0	JEP106 Identity Code [3:0]	RO	`0xb`
[3:0]	PART_1	Part Number [11:8].	RO	`0x2`

6.5.9 PIDR2

The Peripheral ID2 register returns byte[2] of the peripheral ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0E8`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-37: PIDR2 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	**RAZ/WI**	-
[7:4]	REVISION	Revision Code	RO	`0x0`
[3]	JEDEC	JEDEC	RO	`0x1`
[2:0]	DES_1	JEP106 Identity Code [6:4]	RO	`0x3`

6.5.10 PIDR3

The Peripheral ID 3 register returns byte[3] of the peripheral ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0EC`

Type

RO

Default

`0x0000_0000`

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-38: PIDR3 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	**RAZ/WI**	-
[7:4]	REVAND	Manufacturer revision number	RO	0x0
[3:0]	CMOD	Customer Modified	RO	0x0

6.5.11 CIDR0

The Component ID0 register returns byte[0] of the component ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

0x0F0

Type

RO

Default

0x0000_000D

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-39: CIDR0 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	-	Reserved, **RAZ/WI**	-	-
[7:0]	PRMBL_0	Preamble	RO	0xd

6.5.12 CIDR1

The Component ID1 register returns byte[1] of the component ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0F4`

Type

RO

Default

`0x000_000F0`

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-40: CIDR1 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	RAZ/WI	-
[7:4]	CLASS	Component class	RO	`0xf`
[3:0]	PRMBL_1	Preamble	RO	`0x0`

6.5.13 CIDR2

The Component ID2 register returns byte[2] of the component ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

`0x0F8`

Type

RO

Default

`0x0000_0005`

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-41: CIDR2 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	RAZ/WI	-
[7:0]	PRMBL_2	Preamble	RO	0x5

6.5.14 CIDR3

The Component ID3 register returns byte[3] of the component ID.

Configurations

See bit descriptions.

Attributes

Register frame

DMAINFO

Offset

0x0FC

Type

RO

Default

0x0000_00B1

Usage constraints

There are no usage constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-42: CIDR3 register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	RAZ/WI	-
[7:0]	PRMBL_3	Preamble	RO	0xb1

6.6 DMACH<n> summary, DMA Channel Register Frame

This block contains the channel registers related to the execution of the DMA command. When the registers for the command are configured and the ENABLECMD is set, then most of the registers

become read-only except for a few that allow control from the software during the operation of the command.


Note

Certain register fields or registers are stored in each VCH instance, others are stored in each PCH instance where also a shadow copy is stored in the Context memory.

Table 6-43: DMACH<n> register summary

Offset	Name	Type	Default	Width	Description
0x00	CH_CMD	RW	0x0000_0000	32	Stored in VCH instance
0x04	CH_STATUS	RW	0x0000_0000	32	Stored in VCH instance
0x08	CH_INTREN	RW	0x0000_0000	32	Stored in VCH instance
0x0C	CH_CTRL	RW	0x0020_0200	32	CH_CTRL.CHPRIO and CH_CTRL.USE** are stored in VCH instance, all other CH_CTRL fields are stored in PCH instance
0x10	CH_SRCADDR	RW	0x0000_0000	32	Stored in PCH instance
0x14	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x18	CH_DESADDR	RW	0x0000_0000	32	Stored in PCH instance
0x1C	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x20	CH_XSIZE	RW	0x0000_0000	32	Stored in PCH instance
0x24	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x28	CH_SRCTRANSCFG	RW	0x000F_0400	32	Stored in PCH instance
0x2C	CH_DESTRANSCFG	RW	0x000F_0400	32	Stored in PCH instance
0x30	CH_XADDRINC	RW	0x0000_0000	32	Stored in PCH instance
0x34	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x38	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x3C	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x40	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x44	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x48	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x4C	CH_SRCTRIGINCFG	RW	0x0000_0000	32	CH_SRCTRIGINCFG.TYPE and CH_SRCTRIGINCFG.MODE are stored in VCH instance CH_SRCTRIGINCFG.BLKSIZE is stored in PCH instance
0x50	CH_DESTRIGINCFG	RW	0x0000_0000	32	CH_DESTRIGINCF.TYPE and CH_DESTRIGINCF.MODE are stored in VCH instance CH_DESTRIGINCFG.BLKSIZE is stored in PCH instance
0x54	CH_TRIGOUTCFG	RW	0x0000_0000	32	Stored in VCH instance
0x58	CH_GPOEN0	RW	0x0000_0000	32	Stored in VCH instance

Offset	Name	Type	Default	Width	Description
0x60	CH_GPOVAL0	RW	0x0000_0000	32	Stored in VCH instance
0x68	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x70	CH_LINKATTR	RW	0x0000_0000	32	Stored in PCH instance
0x74	CH_AUTOCFG	RW	0x0000_0000	32	CH_AUTOCFG.CMDRESTARTINFE is stored in VCH CH_AUTOCFG.CMDRESTARTCNT is stored in PCH
0x78	CH_LINKADDR	RW	0x0000_0000	32	CH_LINKADDR.LINKADDREN stored in VCH instance. All other fields are stored in PCH instance
0x7C	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x80	CH_GPOREAD0	RO	0x0000_0000	32	-
0x88	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x8C	Reserved	**RAZ/WI**	0x0000_0000	32	-
0x90	CH_ERRINFO	RO	0x0000_0000	32	-
0xC8	CH_IIDR	RO	IMPL_DEF	32	-
0xCC	CH_AIDR	RO	IMPL_DEF	32	-
0xE8	Reserved	**RAZ/WI**	0x0000_0000	32	-
0xF8	CH_BUILDCFG0	RO	IMPL_DEF	32	-
0xFC	CH_BUILDCFG1	RO	IMPL_DEF	32	-

DMA Channel Register Frame global constraints

The following global constraints apply to all registers in this block:

- When the channel is set to Secure, Non-secure accesses to this register block are not allowed and response is based on the security configuration settings.
- When the channel is set to privileged, then unprivileged accesses to this register block are treated as **RAZ/WI**.
- Base address: 0x1000 + 0x0100 * <n>
- Size: 0x0100
- Instances: NUM_CHANNELS

6.6.1 CH_CMD

The Channel DMA Command register allows the software to control the operation of a DMA command.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x000

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-44: CH_CMD register bit descriptions

Bits	Name	Description	Type	Default
[31:25]	Reserved	-	**RAZ/WI**	-
[24]	SWTRIGOUTACK	Software Generated Trigger Output Acknowledge. Write '1' to acknowledge a Trigger Output request from the DMA. Once set to '1', this remains high until the DMA Trigger Output is raised (either on the trigger output signal or as an interrupt) and the acknowledge is accepted. When the channel is not enabled, write to this register is ignored. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	W1S	0x0
[23]	Reserved	-	**RAZ/WI**	-
[22:21]	DESSWTRIGINTYPE	Software Generated Destination Trigger Input Request Type. Selects the trigger request type for the destination trigger input when the software triggers the DESSWTRIGINREQ bit. • 00: Single request • 01: Last single request • 10: Block request • 11: Last block request This field cannot be changed while the DESSWTRIGINREQ bit is set. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	RW	0x0
[20]	DESSWTRIGINREQ	Software Generated Destination Trigger Input Request. Write to '1' to create a software trigger request to the DMA with the specified type in the DESSWTRIGINTYPE register. Once set to '1', this remains high until the DMA is accepted the trigger and returns this to '0'. It is also cleared automatically if the current command is completed without expecting another trigger event. When the channel is not enabled, write to this register is ignored. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	W1S	0x0
[19]	Reserved	-	**RAZ/WI**	-

Bits	Name	Description	Type	Default
[18:17]	SRCSWTRIGINTYPE	Software Generated Source Trigger Input Request Type. Selects the trigger request type for the source trigger input when the software triggers the SRCSWTRIGINREQ bit. • 00: Single request • 01: Last single request • 10: Block request • 11: Last block request This field cannot be changed while the SRCSWTRIGINREQ bit is set. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	RW	0x0
[16]	SRCSWTRIGINREQ	Software Generated Source Trigger Input Request. Write to '1' to create a software trigger request to the DMA with the specified type in the SRCSWTRIGINTYPE register. Once set to '1', this remains high until the DMA accepted the trigger and returns this to '0'. It is also cleared automatically if the current command is completed without expecting another trigger event. When the channel is not enabled, write to this register is ignored. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	W1S	0x0
[15:6]	Reserved	-	**RAZ/WI**	-
[5]	RESUMECMD	Resume Current DMA Operation. Writing this bit to '1' means that the DMAC can continue the operation of a paused channel. Can be set to '1' when the PAUSECMD or a STAT_DONE assertion with DONEPAUSEEN set HIGH results in pausing the current DMA channel operation indicated by the STAT_PAUSED and STAT_RESUMEWAIT bits. Otherwise, writes to this bit are ignored.	W1S	0x0
[4]	PAUSECMD	Pause Current DMA Operation. Once set to '1' the status cannot change until the DMA operation reached the paused state indicated by the STAT_PAUSED and STAT_RESUMEWAIT bits. The bit can be set by software by writing it to '1', the current active DMA operation is paused as soon as possible, but the ENABLECMD bit remains HIGH to show that the operation is still active. Cleared automatically when STAT_RESUMEWAIT is set and the RESUMECMD bit is written to '1', meaning that the software continues the operation of the channel. Note that each DMA channel can have other sources of a pause request and this field does not reflect the state of the other sources. When set at the same time as ENABLECMD or when the channel is not enabled then write to this register is ignored.	W1S	0x0
[3]	STOPCMD	Stop Current DMA Operation. Once set to '1', this remains high until the DMA channel is stopped cleanly. Then this returns to '0' and ENABLECMD is also cleared. When set at the same time as ENABLECMD or when the channel is not enabled then write to this register is ignored. Note that each DMA channel can have other sources of a stop request and this field does not reflect the state of the other sources.	W1S	0x0
[2]	DISABLECMD	Disable DMA Operation at the end of the current DMA command operation. Once set to '1', this field stays high and the current DMA command is allowed to complete, but the DMA does not fetch the next linked command or auto-restarts the DMA command even if they are set. Once the DMA has stopped, it returns to '0' and ENABLECMD is also cleared. When set at the same time as ENABLECMD or when the channel is not enabled then write to this register is ignored.	W1S	0x0
[1]	CLEARCMD	DMA Clear command. When set to '1', it remains high until all DMA channel registers and any internal queues and buffers are cleared, before returning to '0'. When set at the same time as ENABLECMD or while the DMA channel is already enabled, the clear only occurs after any ongoing DMA operation is either completed, stopped or disabled and the ENABLECMD bit is deasserted by the DMA.	W1S	0x0

Bits	Name	Description	Type	Default
[0]	ENABLECMD	Channel Enable. When set to '1', enables the channel to run its programmed task. When set to '1', it cannot be set back to zero, and this field automatically clears to zero when a DMA process is completed. To force the DMA to stop prematurely, you must use CH_CMD.STOPCMD instead.	W1S	0x0

6.6.2 CH_STATUS

The Channel Status register shows the internal status of the DMA command and also reports interrupts about internal events.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x004

Type

RW

Default

0x000_00000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-45: CH_STATUS register bit descriptions

Bits	Name	Description	Type	Default
[31:27]	Reserved	-	RAZ/WI	-
[26]	STAT_TRIGOUTACKWAIT	Channel is waiting for output Trigger Acknowledgement Status. This bit is set to HIGH when DMA starts waiting for output trigger acknowledgment. Automatically cleared when the output trigger acknowledgement is received either from hardware or software source.	RO	0x0
[25]	STAT_DESTRIGINWAIT	Channel is waiting for Destination Trigger Status. This bit is set to HIGH when DMA starts waiting for destination input trigger request. Automatically cleared when the destination trigger request is received either from hardware or software source.	RO	0x0
[24]	STAT_SRCTRIGINWAIT	Channel is waiting for Source Trigger Status. This bit is set to HIGH when DMA starts waiting for source input trigger request. Automatically cleared when the source trigger request is received either from hardware or software source.	RO	0x0

Bits	Name	Description	Type	Default
[23:22]	Reserved	-	**RAZ/WI**	-
[21]	STAT_RESUMEWAIT	Waiting for resume from software Flag. This flag indicates that the DMA channel successfully paused the operation of the command and needs software acknowledgment to resume the operation. Is set to HIGH if STAT_PAUSED is asserted and the PAUSECMD bit set in the command register or when the STAT_DONE is asserted and the DONEPAUSEEN bit is set. Cleared when the RESUMECMD bit is set in the command register.	RO	0x0
[20]	STAT_PAUSED	Paused Status Flag. This flag is set to HIGH if the DMA channel successfully paused the operation of the command. The pause request can come from many internal or external sources. When the request to pause is not asserted anymore the bit is cleared automatically and the command operation can continue.	RO	0x0
[19]	STAT_STOPPED	Stopped Status Flag. This flag is set to HIGH if the DMA channel successfully reached the stopped state. The stop request can come from many internal or external sources. Write '1' to this bit to clear it. Automatically cleared when the ENABLECMD is set.	W1C	0x0
[18]	STAT_DISABLED	Disabled Status Flag. This flag is set to HIGH if the DMA channel is successfully disabled using the DISABLECMD command. Write '1' to this bit to clear it. Automatically cleared when the ENABLECMD is set.	W1C	0x0
[17]	STAT_ERR	Error Status Flag. This flag is set to HIGH if the DMA encounters an error during its operation. The details about the error event can be found in the ERRINFO register. Write '1' to this bit to clear it. When cleared, it also clears the ERRINFO register. Automatically cleared when the ENABLECMD is set.	W1C	0x0
[16]	STAT_DONE	Done Status Flag. This flag is set to HIGH when the DMA command reaches the state defined by the DONETYPE settings. When DONEPAUSEEN is set the DMA command operation is paused when this flag is asserted. Write '1' to this bit to clear it. Automatically cleared when the ENABLECMD is set.	W1C	0x0
[15:11]	Reserved	-	**RAZ/WI**	-
[10]	INTR_TRIGOUTACKWAIT	Channel is waiting for output Trigger Acknowledgment Interrupt Flag. This interrupt is set to HIGH if the INTREN_TRIGOUTACKWAIT is set and the STAT_TRIGOUTACKWAIT status flag is asserted. Automatically cleared when STAT_TRIGOUTACKWAIT is cleared.	RO	0x0
[9]	INTR_DESTRIGINWAIT	Channel is waiting for Destination Trigger Interrupt Flag. This interrupt is set to HIGH if the INTREN_DESTRIGINWAIT is set and the STAT_DESTRIGINWAIT status flag is asserted. Automatically cleared when STAT_DESTRIGINWAIT is cleared.	RO	0x0
[8]	INTR_SRCTRIGINWAIT	Channel is waiting for Source Trigger Interrupt Flag. This interrupt is set to HIGH if the INTREN_SRCTRIGINWAIT is set and the STAT_SRCTRIGINWAIT status flag is asserted. Automatically cleared when STAT_SRCTRIGINWAIT is cleared.	RO	0x0
[7:4]	Reserved	-	**RAZ/WI**	-
[3]	INTR_STOPPED	Stopped Interrupt Flag. This interrupt is set to HIGH if the INTREN_STOPPED is set and the STAT_STOPPED flag gets raised. Automatically cleared when STAT_STOPPED is cleared.	RO	0x0
[2]	INTR_DISABLED	Disabled Interrupt Flag. This interrupt is set to HIGH if the INTREN_DISABLED is set and the STAT_DISABLED flag gets raised. Automatically cleared when STAT_DISABLED is cleared.	RO	0x0
[1]	INTR_ERR	Error Interrupt Flag. This interrupt is set to HIGH if the INTREN_ERR is set and the STAT_ERR status flag gets raised. Automatically cleared when STAT_ERR is cleared.	RO	0x0

Bits	Name	Description	Type	Default
[0]	INTR_DONE	Done Interrupt Flag. This interrupt is set to HIGH if the INTREN_DONE is set and the STAT_DONE status flag gets raised. Automatically cleared when STAT_DONE is cleared.	RO	0x0

6.6.3 CH_INTREN

The Channel Interrupt Enable register can enable the interrupt generation for internal events.

The content of this register can be updated during command linking by setting bit[2] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x008

Type

RW

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-46: CH_INTREN register bit descriptions

Bits	Name	Description	Type	Default
[31:11]	Reserved	-	RAZ/WI	-
[10]	INTREN_TRIGOUTACKWAIT	Channel is waiting for output Trigger Acknowledgement Interrupt Enable. When set to HIGH, enables INTR_TRIGOUTACKWAIT to be set and raise an interrupt when STAT_TRIGOUTACKWAIT status flag is asserted. When set to LOW, it prevents INTR_TRIGOUTACKWAIT to be asserted. Currently pending interrupts are not affected by clearing this bit.	RW	0x0
[9]	INTREN_DESTRIGINWAIT	Channel is waiting for destination Trigger Interrupt Enable. When set to HIGH, enables INTR_DESTRIGINWAIT to be set and raise an interrupt when STAT_DESTRIGINWAIT status flag is asserted. When set to LOW, it prevents INTR_DESTRIGINWAIT to be asserted. Currently pending interrupts are not affected by clearing this bit.	RW	0x0

Bits	Name	Description	Type	Default
[8]	INTREN_SRCTRIGINWAIT	Channel is waiting for source Trigger Interrupt Enable. When set to HIGH, enables INTR_SRCTRIGINWAIT to be set and raise an interrupt when STAT_SRCTRIGINWAIT status flag is asserted. When set to LOW, it prevents INTR_SRCTRIGINWAIT to be asserted. Currently pending interrupts are not affected by clearing this bit.	RW	0x0
[7:4]	Reserved	-	**RAZ/WI**	-
[3]	INTREN_STOPPED	Stopped Interrupt Enable. When set to HIGH, enables INTR_STOPPED to be set and raise an interrupt when STAT_STOPPED status flag is asserted. When set to LOW, it prevents INTR_STOPPED to be asserted. Currently pending interrupts are not affected by clearing this bit.	RW	0x0
[2]	INTREN_DISABLED	Disabled Interrupt Enable. When set to HIGH, enables INTR_DISABLED to be set and raise an interrupt when STAT_DISABLED status flag is asserted. When set to LOW, it prevents INTR_DISABLED to be asserted. Currently pending interrupts are not affected by clearing this bit.	RW	0x0
[1]	INTREN_ERR	Error Interrupt Enable. When set to HIGH, enables INTR_ERROR to be set and raise an interrupt when the STAT_ERR status flag is asserted. When set to LOW, it prevents INTR_ERR to be asserted. Currently pending interrupts are not affected by clearing this bit.	RW	0x0
[0]	INTREN_DONE	Done Interrupt Enable. When set to HIGH, enables the INTR_DONE to be set and raise an interrupt when the STAT_DONE status flag is asserted. When set to LOW, it prevents INTR_DONE to be asserted. Currently pending interrupts are not affected by clearing this bit.	RW	0x0

6.6.4 CH_CTRL

The Channel Control register can be used to configure the type of the transfers and the resources needed by the currently executed DMA command. It also defines how the command shall behave when the command is complete.

The content of this register can be updated during command linking by setting bit[3] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x00C

Type

RW

Default

0x0020_0200

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-47: CH_CTRL register bit descriptions

Bits	Name	Description	Type	Default
[31:29]	Reserved	-	**RAZ/WI**	-
[28]	USEGPO	Enable GPO use for this command. • 0: disable • 1: enable The field is **RAZ/WI** when the following condition is False: GPO_WIDTH > 0.	RW	0x0
[27]	USETRIGOUT	Enable Trigger Output use for this command. • 0: disable • 1: enable	RW	0x0
[26]	USEDESTRIGIN	Enable Destination Trigger Input use for this command.e • 0: disable • 1: enable	RW	0x0
[25]	USESRCTRIGIN	Enable Source Trigger Input use for this command: • 0: disable • 1: enable	RW	0x0
[24]	DONEPAUSEEN	Done pause enable. When set to HIGH the assertion of the STAT_DONE flag results in an automatic pause request for the current DMA operation. When the paused state is reached the STAT_RESUMEWAIT flag is also set. When set to LOW the assertion of the STAT_DONE does not pause the progress of the command and the next operation of the channel is started immediately after the STAT_DONE flag is set.	RW	0x0
[23]	Reserved	-	**RAZ/WI**	-
[22:21]	DONETYPE	Done type selection. This field defines when the STAT_DONE status flag is asserted during the command operation. • 00: STAT_DONE flag is not asserted for this command. • 01: End of a command, before jumping to the next linked command. (default) • 10: Reserved. • 11: End of an autorestart cycle, before starting the next cycle.	RW	0x1

Bits	Name	Description	Type	Default
[20:18]	REGRELOADTYPE	Automatic register reload type. Defines how the DMA command reloads initial values at the end of a DMA command before autorestarting, ending or linking to a new DMA command: • 000: Reload Disabled. • 001: Reload source and destination size registers only. • 011: Reload source address only and all source and destination size registers. • 101: Reload destination address only and all source and destination size registers. • 111: Reload source and destination address and all source and destination size registers. • Others: Reserved. Note: When CLEARCMD is set, the reloaded registers are also cleared.	RW	0x0
[17:12]	Reserved	-	**RAZ/WI**	-
[11:9]	XTYPE	Operation type for X direction: • 000: disable - No data transfer takes place for this command. This mode can be used to create empty commands that wait for an event or set GPOs. • 001: continue - Copy data in a continuous manner from source to the destination. For 1D operations it is expected that SRCXSIZE is equal to DESXSIZE, other combinations result in **UNPREDICTABLE** behavior. • Others: Reserved.	RW	0x1
[8]	CHLOCKREQ	Channel Lock Request. When set to '1' the DMA locks this channel to physical channel. This field is ignored when the number of channels match the number of physical channels.	RW	0x0
[7:4]	CHPRIO	Channel Priority. • 0: Lowest Priority • 15: Highest Priority	RW	0x0
[3]	Reserved	-	**RAZ/WI**	-
[2:0]	TRANSIZE	Transfer Entity Size. Size in bytes = 2^TRANSIZE. • 000: Byte • 001: Halfword • 010: Word • 011: Doubleword • 100: 128bits • 101: 256bits • 110: 512bits • 111: 1024bits Note: Setting a value greater than the configured DATA_WIDTH results in a configuration error (CFGERR).	RW	0x0

6.6.5 CH_SRCADDR

The Channel Source Address register defines the 32-bit base address of the command to read the data from. When the register is read during the execution of the command it shows an approximate hint at the actual address the DMA channel is going to read from.

The content of this register can be updated during command linking by setting bit[4] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x010

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-48: CH_SRCADDR register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	SRCADDR	Source Address [31:0].	RW	0x0

6.6.6 CH_DESADDR

The Channel Destination Address register defines the 32-bit base address of the command to write the data to. When the register is read during the execution of the command it shows an approximate hint at the actual address the DMA channel is going to write to.

The content of this register can be updated during command linking by setting bit[6] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x018

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-49: CH_DESADDR register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	DESADDR	Destination Address[31:0]	RW	0x0

6.6.7 CH_XSIZE

The Channel X Dimension Size Register, Lower Bits [15:0] register defines the number of data units copied during the DMA command up to 16 bits in the X dimension. The source and destination size of the command may be different for some transfer types. When read during the execution of the DMA command, the register shows an approximate hint of the remaining number of lines in both read and write directions.

The content of this register can be updated during command linking by setting bit[8] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x020

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-50: CH_XSIZE register bit descriptions

Bits	Name	Description	Type	Default
[31:16]	DESXSIZE	Destination Number of Transfers in the X Dimension lower bits [15:0]. This register defines the destination data block size of the DMA operation for or any 1D operation.	RW	0x0
[15:0]	SRCXSIZE	Source Number of Transfers in the X Dimension lower bits [15:0]. This register defines the source data block size of the DMA operation for any ID operation.	RW	0x0

6.6.8 CH_SRCTRANSCFG

The Channel Source Transfer Configuration register provides transfer attribute settings in the read direction of the DMA command.

The content of this register can be updated during command linking by setting bit[10] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x028

Type

RW

Default

0x000F_0400

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-51: CH_SRCTRANSCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:20]	Reserved	-	**RAZ/WI**	-
[19:16]	SRCMAXBURSTLEN	Source Max Burst Length. Hint for the DMA on what is the maximum allowed burst size it can use for read transfers. The maximum number of beats sent by the DMA for a read burst is equal to SRCMAXBURSTLEN + 1. Default value is 16 beats, which allows the DMA to set all burst sizes. Note: Limited by the DATA_BUFF_SIZE so larger settings may not always result in larger bursts.	RW	`0xf`
[15:12]	Reserved	-	**RAZ/WI**	-
[11]	SRCPRIVATTR	Source Transfer Privilege Attribute. • 0: Unprivileged • 1: Privileged When a channel is unprivileged this bit is tied to 0.	RW	`0x0`
[10]	SRCNONSECATTR	Source Transfer Non-secure Attribute. • 0: Secure • 1: Non-secure When a channel is Non-secure this bit is tied to 1.	RW	`0x1`
[9:8]	SRCSHAREATTR	Source Transfer Shareability Attribute. • 00: Non-shareable • 01: Reserved • 10: Outer shareable • 11: Inner shareable	RW	`0x0`

Bits	Name	Description	Type	Default
[7:4]	SRCMEMATTRHI	Source Transfer Memory Attribute field [7:4]. • 0000: Device memory • 0001: Normal memory, Outer Write allocate, Outer Write -through transient • 0010: Normal memory, Outer Read allocate, Outer Write-through transient • 0011: Normal memory, Outer Read/Write allocate, Outer Write-through transient • 0100: Normal memory, Outer non-cacheable • 0101: Normal memory, Outer Write allocate, Outer Write-back transient • 0110: Normal memory, Outer Read allocate, Outer Write-back transient • 0111: Normal memory, Outer Read/Write allocate, Outer Write-back transient • 1000: Normal memory, Outer Write-through non-transient • 1001: Normal memory, Outer Write allocate, Outer Write-through non-transient • 1010: Normal memory, Outer Read allocate, Outer Write-through non-transient • 1011: Normal memory, Outer Read/Write allocate, Outer Write-through non-transient • 1100: Normal memory, Outer Write-back non-transient • 1100: Normal memory, Outer Write-back non-transient • 1101: Normal memory, Outer Write allocate, Outer Write-back non-transient • 1110: Normal memory, Outer Read allocate, Outer Write-back non-transient • 1111: Normal memory, Outer Read/Write allocate, Outer Write-back non-transient	RW	0x0

Bits	Name	Description	Type	Default
[3:0]	SRCMEMATTRLO	Source Transfer Memory Attribute field [3:0]. When SRCMEMATTRHI is Device type (0000) then this field means: • 0000: Device-nGnRnE • 0100: Device-nGnRE • 1000: Device-nGRE • 1100: Device-GRE • Others: Invalid resulting in **UNPREDICTABLE** behavior. When SRCMEMATTRHI is Normal memory type (other than 0000) then this field means: • 0000: Reserved • 0001: Normal memory, Inner Write allocate, Inner Write-through transient • 0010: Normal memory, Inner Read allocate, Inner Write-through transient • 0011: Normal memory, Inner Read/Write allocate, Inner Write-through transient • 0100: Normal memory, Inner non-cacheable • 0101: Normal memory, Inner Write allocate, Inner Write-back transient • 0110: Normal memory, Inner Read allocate, Inner Write-back transient • 0111: Normal memory, Inner Read/Write allocate, Inner Write-back transient • 1000: Normal memory, Inner Write-through non-transient • 1001: Normal memory, Inner Write allocate, Inner Write-through non-transient • 1010: Normal memory, Inner Read allocate, Inner Write-through non-transient • 1011: Normal memory, Inner Read/Write allocate, Inner Write-through non-transient • 1011: Normal memory, Inner Read/Write allocate, Inner Write-through non-transient • 1100: Normal memory, Inner Write-back non-transient • 1101: Normal memory, Inner Write allocate, Inner Write-back non-transient • 1110: Normal memory, Inner Read allocate, Inner Write-back non-transient • 1111: Normal memory, Inner Read/Write allocate, Inner Write-back non-transient	RW	0x0

6.6.9 CH_DESTRANSCFG

The Channel Destination Transfer Configuration register provides transfer attribute settings in the write direction of the DMA command.

The content of this register can be updated during command linking by setting bit[11] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x02C

Type

RW

Default

0x000F_0400

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-52: CH_DESTRANSCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:20]	Reserved	-	RAZ/WI	-
[19:16]	DESMAXBURSTLEN	Destination Max Burst Length. Hint for the DMA on what is the maximum allowed burst size it can use for write transfers. The maximum number of beats sent by the DMA for a write burst is equal to DESMAXBURSTLEN + 1. Default value is 16 beats, which allows the DMA to set all burst sizes. Note: Limited by the DATA_BUFF_SIZE so larger settings may not always result in larger bursts.	RW	0xf
[15:12]	Reserved	-	RAZ/WI	-
[11]	DESPRIVATTR	Destination Transfer Privilege Attribute. • 0: Unprivileged • 1: Privileged When a channel is unprivileged this bit is tied to 0.	RW	0x0
[10]	DESNONSECATTR	Destination Transfer Non-secure Attribute. • 0: Secure • 1: Non-secure When a channel is Non-secure this bit is tied to 1.	RW	0x1
[9:8]	DESSHAREATTR	Destination Transfer Shareability Attribute. • 00: Non-shareable • 01: Reserved • 10: Outer shareable • 11: Inner shareable	RW	0x0

Bits	Name	Description	Type	Default
[7:4]	DESMEMATTRHI	Destination Transfer Memory Attribute field [7:4]. • 0000: Device memory • 0001: Normal memory, Outer Write allocate, Outer Write-through transient • 0010: Normal memory, Outer Read allocate, Outer Write-through transient • 0011: Normal memory, Outer Read/Write allocate, Outer Write-through transient • 0100: Normal memory, Outer non-cacheable • 0101: Normal memory, Outer Write allocate, Outer Write-back transient • 0110: Normal memory, Outer Read allocate, Outer Write-back transient • 0111: Normal memory, Outer Read/Write allocate, Outer Write-back transient • 1000: Normal memory, Outer Write-through non-transient • 1001: Normal memory, Outer Write allocate, Outer Write-through non-transient • 1010: Normal memory, Outer Read allocate, Outer Write-through non-transient • 1011: Normal memory, Outer Read/Write allocate, Outer Write-through non-transient • 1100: Normal memory, Outer Write-back non-transient • 1101: Normal memory, Outer Write allocate, Outer Write-back non-transient • 1110: Normal memory, Outer Read allocate, Outer Write-back non-transient • 1111: Normal memory, Outer Read/Write allocate, Outer Write-back non-transient	RW	0x0

Bits	Name	Description	Type	Default
[3:0]	DESMEMATTRLO	Destination Transfer Memory Attribute field [3:0]. When DESMEMATTRHI is Device type (0000) then this field means: • 0000: Device-nGnRnE • 0100: Device-nGnRE • 1000: Device-nGRE • 1100: Device-GRE • Others: Invalid resulting in **UNPREDICTABLE** behavior When DESMEMATTRHI is Normal Memory type (other than 0000) then this field means: • 0000: Reserved • 0001: Normal memory, Inner Write allocate, Inner Write-through transient • 0010: Normal memory, Inner Read allocate, Inner Write-through transient • 0011: Normal memory, Inner Read/Write allocate, Inner Write-through transient • 0100: Normal memory, Inner non-cacheable • 0101: Normal memory, Inner Write allocate, Inner Write-back transient • 0110: Normal memory, Inner Read allocate, Inner Write-back transient • 0111: Normal memory, Inner Read/Write allocate, Inner Write-back transient • 1000: Normal memory, Inner Write-through non-transient • 1001: Normal memory, Inner Write allocate, Inner Write-through non-transient • 1010: Normal memory, Inner Read allocate, Inner Write-through non-transient • 1011: Normal memory, Inner Read/Write allocate, Inner Write-through non-transient • 1100: Normal memory, Inner Write-back non-transient • 1101: Normal memory, Inner Write allocate, Inner Write-back non-transient • 1110: Normal memory, Inner Read allocate, Inner Write-back non-transient • 1111: Normal memory, Inner Read/Write allocate, Inner Write-back non-transient	RW	0x0

6.6.10 CH_XADDRINC

The Channel X Dimension Address Increment register sets the increment values used to update the source and destination addresses after each transferred data unit.

The content of this register can be updated during command linking by setting bit[12] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x030

Type

RW

Default

0x00000000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-53: CH_XADDRINC register bit descriptions

Bits	Name	Description	Type	Default
[31:16]	DESXADDRINC	Destination X dimension Address Increment. This value is used as the increment between each TRANSIZE transfer. When a single bit is used then only 0 and 1 can be set. For wider increment registers, two's complement used with a range between -32768 to 32767 when the counter is 16 bits wide. The width of the register is indicated by the INC_WIDTH parameter. DESADDR_next = DESADDR + 2^TRANSIZE * DESXADDRINC	RW	0x0
[15:0]	SRCXADDRINC	Source X dimension Address Increment. This value is used as the increment between each TRANSIZE transfer. When a single bit is used then only 0 and 1 can be set. For wider increment registers, two's complement used with a range between -32768 to 32767 when the counter is 16 bits wide. The width of the register is indicated by the INC_WIDTH parameter. SRCADDR_next = SRCADDR + 2^TRANSIZE * SRCXADDRINC	RW	0x0

6.6.11 CH_SRCTRIGINCFG

The Channel Source Trigger In Configuration register provides configuration settings when using source side trigger input for the current DMA command.

The content of this register can be updated during command linking by setting bit[19] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x04C

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-54: CH_SRCTRIGINCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:24]	Reserved	-	**RAZ/WI**	-
[23:16]	SRCTRIGINBLKSIZE	Source Trigger Input Default Transfer Size. Defined transfer size per trigger + 1. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	RW	0x0
[15:12]	Reserved		**RAZ/WI**	-
[11:10]	SRCTRIGINMODE	Source Trigger Input Mode: • 00: Command • 01: Reserved • 10: DMA driven Flow control. Only allowed when HAS_TRIGIN is enabled. • 11: Peripheral driven Flow control. Only allowed when HAS_TRIGIN is enabled. Note: This field is ignored for Internal triggers as they only support Command triggers.	RW	0x0
[9:8]	SRCTRIGINTYPE	Source Trigger Input Type: • 00: Software only Trigger Request. SRCTRIGINSEL is ignored. • 01: Reserved • 10: hardware Trigger Request. Only allowed when HAS_TRIGIN is enabled. SRCTRIGINSEL selects between external trigger inputs if HAS_TRIGSEL is enabled. • 11: Internal Trigger Request. Only allowed when HAS_TRIGSEL is enabled and the DMAC has multiple channels, otherwise treated as hardware Trigger Request. SRCTRIGINSEL selects between DMA channels. Note: software triggers are also available when hardware or Internal types are selected, but is is not recommended and caution must be taken when the these modes are combined.	RW	0x0
[7:0]	Reserved	-	**RAZ/WI**	-

6.6.12 CH_DESTRIGINCFG

The Channel Destination Trigger In Configuration register provides configuration settings when using destination side trigger input for the current DMA command.

The content of this register can be updated during command linking by setting bit[20] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x050

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also Global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-55: CH_DESTRIGINCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:24]	Reserved	-	**RAZ/WI**	-
[23:16]	DESTRIGINBLKSIZE	Destination Trigger Input Default Transfer Size. Defined transfer size per trigger + 1. The field is **RAZ/WI** when the following condition is False: HAS_TRIG	RW	0x0
[15:12]	Reserved	-	**RAZ/WI**	-
[11:10]	DESTRIGINMODE	Destination Trigger Input Mode: • 00: Command • 01: Reserved • 10: DMA driven Flow control. Only allowed when HAS_TRIGIN is enabled. • 11: Peripheral driven Flow control. Only allowed when HAS_TRIGIN is enabled. Note: This field is ignored for Internal triggers as they only support Command triggers.	RW	0x0

Bits	Name	Description	Type	Default
[9:8]	DESTRIGINTYPE	Destination Trigger Input Type: • 00: Software only Trigger Request. DESTRIGINSEL is ignored • 01: Reserved • 10: Hardware Trigger Request. Only allowed when HAS_TRIGIN is enabled. DESTRIGINSEL selects between external trigger inputs if HAS_TRIGSEL is enabled. • 11: Internal Trigger Request. Only allowed when HAS_TRIGSEL is enabled and the DMAC has multiple channels, otherwise treated as hardware Trigger Request. DESTRIGINSEL selects between DMA channels. Note: Software triggers are also available when hardware or Internal types are selected, but is is not recommended and caution must be taken when the these modes are combined.	RW	0x0
[7:0]	Reserved	-	RAZ/WI	-

6.6.13 CH_TRIGOUTCFG

The Channel Trigger Out Configuration register provides configuration settings when using trigger output for the current DMA command.

The content of this register can be updated during command linking by setting bit[21] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x054

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-56: CH_TRIGOUTCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:10]	Reserved	-	**RAZ/WI**	-
[9:8]	TRIGOUTTYPE	Trigger Output Type • 00: Software only Trigger Acknowledgment. • 01: Reserved • 10: Hardware Trigger Acknowledgment. Only allowed when HAS_TRIGOUT is enabled. • 11: Internal Trigger Acknowledgment. Only allowed when HAS_TRIGSEL is enabled and the DMAC has multiple channels, otherwise treated as Hardware Trigger Acknowledgment.	RW	0x0
[7:0]	Reserved	-	**RAZ/WI**	-

6.6.14 CH_GPOEN0

The Channel GPO Driving Enable register 0 enables which GPO ports are enabled to change at the beginning of current DMA command. GPO ports from bit 0 to 31 can be enabled by this register if the port is available to this channel.

The content of this register can be updated during command linking by setting bit[22] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x058

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-57: CH_GPOEN0 register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	GPOEN0	Channel General Purpose Output (GPO) bit 0 to 31 enable mask. If bit n is '1', then GPO[n] is selected for driving by GPOVAL0[n]. If bit 'n' is '0', then GPO[n] keeps its previous value. Only [GPO_WIDTH-1:0] are implemented. All unimplemented bits are **RAZ/WI**. The field is **RAZ/WI** when the following condition is False: GPO_WIDTH > 0	RW	0x0

6.6.15 CH_GPOVAL0

The Channel GPO Value register 0 sets the value to be driven on the GPO ports that are enabled at the beginning of current DMA command. The value of the GPO ports from bit 0 to 31 can be adjusted by this register if the port is available to this channel.

The content of this register can be updated during command linking by setting bit[24] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x060

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-58: CH_GPOVAL0 register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	GPOVAL0	General Purpose Output Value GPO[31:0]. Write to set output value. The actual value on the GPO port becomes active when the command is enabled. Read returns the register value which might be different from the actual GPO port status. Only [GPO_WIDTH-1:0] are implemented. All unimplemented bits are **RAZ/WI**. The field is **RAZ/WI** when the following condition is False: GPO_WIDTH > 0	RW	0x0

6.6.16 CH_LINKATTR

The Channel Link Address Memory Attributes register provides transfer attribute settings for the command link related read transfers. The security and privilege attributes cannot be adjusted, they match the attributes of the channel.

The content of this register can be updated during command linking by setting bit[28] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x070

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-59: CH_LINKATTR register bit descriptions

Bits	Name	Description	Type	Default
[31:10]	Reserved	-	**RAZ/WI**	-

Bits	Name	Description	Type	Default
[9:8]	LINKSHAREATTR	Link Address Transfer Shareability Attribute. • 00: Non-shareable • 01: Reserved • 10: Outer shareable • 11: Inner shareable	RW	0x0
[7:4]	LINKMEMATTRHI	Link Address Read Transfer Memory Attribute field [7:4]. • 0000: Device memory • 0001: Normal memory, Outer Write allocate, Outer Write-through transient • 0010: Normal memory, Outer Read allocate, Outer Write-through transient • 0011: Normal memory, Outer Read/Write allocate, Outer Write-through transient • 0100: Normal memory, Outer non-cacheable • 0101: Normal memory, Outer Write allocate, Outer Write-back transient • 0110: Normal memory, Outer Read allocate, Outer Write-back transient • 0111: Normal memory, Outer Read/Write allocate, Outer Write-back transient • 1000: Normal memory, Outer Write-through non-transient • 1001: Normal memory, Outer Write allocate, Outer Write-through non-transient • 1010: Normal memory, Outer Read allocate, Outer Write-through non-transient • 1011: Normal memory, Outer Read/Write allocate, Outer Write-through non-transient • 1100: Normal memory, Outer Write-back non-transient • 1101: Normal memory, Outer Write allocate, Outer Write-back non-transient • 1110: Normal memory, Outer Read allocate, Outer Write-back non-transient • 1111: Normal memory, Outer Read/Write allocate, Outer Write-back non-transient	RW	0x0

Bits	Name	Description	Type	Default
[3:0]	LINKMEMATTRLO	Link Address Read Transfer Memory Attribute field [3:0]. When LINKMEMATTRHI is Device type (0000) then this field means: • 0000: Device-nGnRnE • 0100: Device-nGnRE • 1000: Device-nGRE • 1100: Device-GRE • Others: Invalid resulting in **UNPREDICTABLE** behavior When LINKMEMATTRHI is Normal memory type (other than 0000) then this field means: • 0000: Reserved • 0001: Normal memory, Inner Write allocate, Inner Write-through transient • 0010: Normal memory, Inner Read allocate, Inner Write-through transient • 0011: Normal memory, Inner Read/Write allocate, Inner Write-through transient • 0100: Normal memory, Inner non-cacheable • 0101: Normal memory, Inner Write allocate, Inner Write-back transient • 0110: Normal memory, Inner Read allocate, Inner Write-back transient • 0111: Normal memory, Inner Read/Write allocate, Inner Write-back transient • 1000: Normal memory, Inner Write-through non-transient • 1001: Normal memory, Inner Write allocate, Inner Write-through non-transient • 1010: Normal memory, Inner Read allocate, Inner Write-through non-transient • 1011: Normal memory, Inner Read/Write allocate, Inner Write-through non-transient • 1100: Normal memory, Inner Write-back non-transient • 1101: Normal memory, Inner Write allocate, Inner Write-back non-transient • 1110: Normal memory, Inner Read allocate, Inner Write-back non-transient • 1111: Normal memory, Inner Read/Write allocate, Inner Write-back non-transient	RW	0x0

6.6.17 CH_AUTOCFG

The Channel Automatic Command Restart Configuration register configures the automatic restart behavior of the currently running command.

The content of this register can be updated during command linking by setting bit[29] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x074

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-60: CH_AUTOCFG register bit descriptions

Bits	Name	Description	Type	Default
[31:17]	Reserved	-	RAZ/WI	-
[16]	CMDRESTARTINFEN	Enable Infinite Automatic Command Restart. When set, CMDRESTARTCNT is ignored and the command is always restarted after it is completed, including output triggering if enabled and autoreloading the registers but it does not perform a link to the next command. This means that the infinite loop of automatic restarts can only be broken by DISABLECMD or STOPCMD. When CMDRESTARTINFEN is set to '0', then the autorestarting of a command depends on CMDRESTARTCNT and when that counter is set to 0 the autorestarting is finished. In this case the next linked command is read or the command is complete.	RW	0x0
[15:0]	CMDRESTARTCNT	Automatic Command Restart Counter. Defines the number of times automatic restarting occurs at end of DMA command. Auto restarting occurs after the command is completed, including output triggering if enabled and autoreloading the registers, but it only performs a link to the next command when CMDRESTARTCNT == 0. When CMDRESTARTCNT and CMDRESTARTINF are both set to '0', autorestart is disabled.	RW	0x0

6.6.18 CH_LINKADDR

The Channel Link Address register sets the 32-bit address of the next element in a command link. When the command execution is finished and this register is set then the DMA channel starts loading the next command from this address.

The content of this register can be updated during command linking by setting bit[30] in the command link header.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x078

Type

RW

Default

0x0000_0000

Usage constraints

Becomes read-only after ENABLECMD is set.

There are also global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-61: CH_LINKADDR register bit descriptions

Bits	Name	Description	Type	Default
[31:2]	LINKADDR	Link Address Pointer [31:2]. The DMAC fetches the next command from this address if LINKADDREN is set. NOTE: Commands are fetched with the security and privilege attribute of the channel and cannot be adjusted for the command link reads.	RW	0x0
[1]	Reserved	-	RAZ/WI	-
[0]	LINKADDREN	Enable Link Address. When set to '1', the DMAC fetches the next command defined by LINKADDR. When set to '0', the DMAC returns to idle at the end of the current command. NOTE: the linked command fetched by the DMAC needs to clear this field to mark the end of the command chain. Otherwise it may result in an infinite loop of the same command.	RW	0x0

6.6.19 CH_GPOREAD0

The Channel GPO Read Value register 0 shows the current value of the GPO ports from bit 0 to 31 that are available to this channel.

The values here can be different from the values driven to the GPO ports by this channel.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x080

Type

RO

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-62: CH_GPOREAD0 register bit descriptions

Bits	Name	Description	Type	Default
[31:0]	GPOREAD0	General Purpose Output Read Value for GPO[31:0]. Read returns the actual value of the GPO ports. Only [GPO_WIDTH-1:0] are implemented. All unimplemented bits are set to 0. The field is **RAZ/WI** when the following condition is False: GPO_WIDTH > 0	RO	0x0

6.6.20 CH_ERRINFO

The Channel Error Information register provides information about internal errors encountered during command execution by the DMA channel.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x090

Type

RO

Default

0x0000_0000

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-63: CH_ERRINFO register bit descriptions

Bits	Name	Description	Type	Default
[31:16]	ERRINFO	Error information Register. Additional information for the error that is encountered by the DMA. The meaning of the bits are detailed in the Error Handling section.	RO	0x0

Bits	Name	Description	Type	Default
[15:2]	Reserved	-	RAZ/WI	-
[1]	CFGERR	Configuration Error Flag. Set when the DMA command is enabled or a linked command is read but it is configured in a mode that is not supported by the implementation.	RO	0x0
[0]	BUSERR	Bus Error Flag. Set when the DMA encounters a bus error during data or command read transfers.	RO	0x0

6.6.21 CH_IIDR

The Channel Implementation Identification register provides information about the revision of the product implementing the DMA channel.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x0C8

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-64: CH_IIDR register bit descriptions

Bits	Name	Description	Type	Default
[31:20]	PRODUCTID	Indicates the product ID	RO	0x3a0
[19:16]	VARIANT	Indicates the major revision, or variant, of the product rxpy identifier	RO	0x0
[15:12]	REVISION	Indicates the minor revision of the product rxpy identifier	RO	0x0
[11:0]	IMPLEMENTER	Contains the JEP106 code of the company that implemented the IP: • [11:8] - JEP106 continuation code of implementer. • [7] - Always 0. • [6:0] - JEP106 identity code of implementer For Arm this field reads as 0x43B.	RO	0x43b

6.6.22 CH_AIDR

The Channel Architecture Identification register provides information about the architecture version supported by the DMA channel.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

0x0CC

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints apart from Global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-65: CH_AIDR register bit descriptions

Bits	Name	Description	Type	Default
[31:8]	Reserved	-	**RAZ/WI**	-
[7:4]	ARCH_MAJOR_REV	Architecture Major Revision	RO	0x0
[3:0]	ARCH_MINOR_REV	Architecture Minor Revision	RO	0x0

6.6.23 CH_BUILDCFG0

The Channel Build Configuration and Capability register 0 contains the configuration parameters and capabilities of the DMA channel.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

`0x0F8`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-66: CH_BUILDCFG0 register bit descriptions

Bits	Name	Description	Type	Default
[31:30]	Reserved	-	**RAZ/WI**	-
[29:26]	INC_WIDTH	Width of the increment register = INC_WIDTH + 1. When set to 0, then only 0 and 1 can be set as an increment. When larger values are used, then negative increments can be set using 2's complement: • INC_WIDTH = 0: 0..1 • INC_WIDTH = 1: -2..1 • INC_WIDTH = 2: -4..3 • INC_WIDTH = 3: -8..7 • ... • INC_WIDTH = 15: -32768..32767	RO	`0xf`
[25]	Reserved	-	**RAZ/WI**	-
[24:22]	DATA_WIDTH	Data Width: • 000: 8-bit • 001: 16-bit • 010: 32-bit • 011: 64-bit • 100: 128-bit • 101: 256-bit • 110: 512-bit • 111: 1024-bit	RO	log2(DATA_WIDTH / 8)
[21:16]	ADDR_WIDTH	Address Width in bits = ADDR_WIDTH + 1	RO	31 |
[15:8]	CMD_BUFF_SIZE	Command Buffer Size in command words - 1. Total Size = (CMD_BUFF_SIZE + 1) * 4 bytes.	RO	0
[7:0]	DATA_BUFF_SIZE	Data Buffer Size in entries - 1. Total Size = ((DATA_BUFF_SIZE + 1) * DATA_WIDTH / 8) bytes.	RO	FIFO_DEPTH - 1 |

6.6.24 CH_BUILDCFG1

The Channel Build Configuration and Capability register 1 contains the configuration parameters and capabilities of the DMA channel.

Configurations

See bit descriptions.

Attributes

Register frame

DMACH<n>

Offset

`0x0FC`

Type

RO

Default

IMPLEMENTATION DEFINED

Usage constraints

There are no usage constraints apart from global constraints.

Bit descriptions

The following table shows the register bit assignments.

Table 6-67: CH_BUILDCFG1 register bit descriptions

Bits	Name	Description	Type	Default
[31:26]	Reserved	-	**RAZ/WI**	-
[25:19]	GPO_WIDTH	General Purpose Output Width. 0 to 64. The field is **RAZ/WI** when the following condition is False: GPO_WIDTH > 0	RO	GPO_WIDTH
[18]	HAS_GPOSEL	Has shared GPO ports between channels.	RO	`0x0`
[17:13]	Reserved	-	**RAZ/WI**	-
[12]	HAS_STREAMSEL	Has selectable stream interfaces.	RO	`0x0`
[11]	HAS_STREAM	Has stream interface support.	RO	`0x0`
[10]	HAS_WRKREG	Has internal register view capability.	RO	`0x0`
[9]	HAS_AUTO	Has automatic reload and restart capability.	RO	`0x1`
[8]	HAS_CMDLINK	Has command link list capability.	RO	`0x1`
[7]	HAS_TRIGSEL	Has selectable triggers.	RO	`0x0`
[6]	HAS_TRIGOUT	Has hardware trigger output port.	RO	HAS_TRIG
[5]	HAS_TRIGIN	Has hardware trigger input ports.	RO	HAS_TRIG
[4]	HAS_TRIG	Has trigger capability.	RO	HAS_TRIG

Bits	Name	Description	Type	Default
[3]	HAS_TMPLT	Has template based pack and unpack capability.	RO	0x0
[2]	HAS_2D	Has 2D capability.	RO	0x0
[1]	HAS_WRAP	Has wrap capability.	RO	0x0
[0]	HAS_XSIZEHI	Has 32-bit XSIZE counters.	RO	0x0

Appendix A Signal descriptions

This appendix contains all signal descriptions.

Clock and Reset

Connection

Clock and reset controller

Table A-1: Clock and reset signals

Signal	Direction	Description
clk	Input	Clock input.
resetn	Input	Active-LOW reset. Reset can go LOW asynchronously but must go HIGH synchronously.
pclken	Input	APB5 Clock enable. inputs sampled when HIGH, outputs are stable when LOW.

LPI Q-Channel signals

Connection

Clock controller

Table A-2: Q-Channel signals

Signal	Direction	Description
clk_qreqn	Input	This signal indicates when the clock controller issues a clock quiescence entry or exit request to the DMAC.
clk_qacceptn	Output	This signal indicates when the DMAC accepts the clock quiescence request.
clk_qdeny	Output	This signal indicates when the DMAC denies the clock quiescence request.
clk_qactive	Output	This signal indicates when the DMAC is active and also when it requests to exit from clock quiescence.

LPI P-Channel signals

Connection

Power controller

Table A-3: P-Channel signals

Signal	Direction	Description
preq	Input	This signal indicates when the power controller issues a power state change request to the DMAC.
pstate[3:0]	Input	This signal indicates the requested power state for the DMAC.
paccept	Output	This signal indicates when the DMAC accepts the power state change request.
pdeny	Output	This signal indicates when the DMAC denies the power state change request.
pactive[9:0]	Output	This signal indicates which DMAC power states are active. The following bits can be asserted by the DMAC: • [8] - ON state • [5] – FULL_RET state All other bits are tied to 0.

APB5 signals

Connection

APB5 interconnect

Table A-4: APB5 signals

Signal	Direction	Description
psel	Input	Select. It indicates that the subordinate device is selected and that a data transfer is required.
penable	Input	Enable. This signal indicates the second and subsequent cycles of an APB5 transfer.
pprot[2:0]	Input	Protection type. This signal indicates the normal, privileged, or Secure protection level of the transaction and whether the transaction is a data access or an instruction access.
pwrite	Input	Direction. This signal indicates an APB5 Write-Access when HIGH and an APB5 read access when LOW.
paddr[13:0]	Input	Address. This is the APB5 address bus.
pwdata[31:0]	Input	Write data. This bus is driven during write cycles when pwrite is HIGH.
pstrb[3:0]	Input	Write strobes. This signal indicates which byte lanes to update during a write transfer. Write strobes must not be active during a read transfer. Note that individual byte lane update is not supported, that is, pstrb is expected to be either `0b0` or `0b1`, otherwise an error response is sent.
pready	Output	Ready. The subordinate uses this signal to extend an APB5 transfer.
pslverr	Output	This signal indicates a transfer failure.
prdata[31:0]	Output	Read Data. The selected subordinate drives this bus during read cycles when pwrite is LOW.
pwakeup	Input	Pending APB5 activity indicator, used as an input to clock and power controllers to wake the DMAC up.
pdebug	Input	Additional sideband signal to identify debugger access. Valid together with psel.

Data and Context AHB5 signals

Connection

AHB5 interconnect

Table A-5: DATA AHB5 signals

Signal	Direction	Description
haddr_dmaif[31:0]	Output	The byte address of the transfer indicator.
hburst_dmaif[2:0]	Output	Indicates how many transfers are in the burst.
hmastlock_dmaif	Output	Indicates that the current transfer is part of a locked sequence. The DMA doesn't initiate locked transfers, therefore it ties this signal to LOW
hprot_dmaif[6:0]	Output	Protection control signal, which provides information about the access type.
hsize_dmaif[2:0]	Output	Indicates the size of the transfer.
hnonsec_dmaif	Output	Indicates whether the transfer is Secure or Non-secure. Not present when SECEXT_PRESENT is set to 0.
htrans_dmaif[1:0]	Output	Indicates the transfer type.
hwdata_dmaif[DATA_WIDTH-1:0]	Output	Transfers data from the Manager to the Subordinate during write operations.
hwstrb_dmaif[DATA_WIDTH/8-1:0]	Output	Write strobes. The DMA does not initiate sparse write operations, therefore it drives hwstrb HIGH for active byte lanes during the data phase of write operations. Sparse writes are not supported
hwrite_dmaif	Output	Indicates the transfer direction.
hrdata_dmaif[DATA_WIDTH-1:0]	Input	Read data bus.

Signal	Direction	Description
hready_dmaif	Input	When HIGH, the hready signal indicates to the Manager and all Subordinates, that the previous transfer is complete.
hresp_dmaif	Input	Transfer response
hauser_dmaif[CHID_WIDTH + NUM_VCH_WIDTH:0] where NUM_VCH_WIDTH = log2(NUM_VCH) When CHID_WIDTH parameter is 0, this port is not present	Output	Address-phase sideband signals mapped as: • [hauser[CHID_WIDTH + NUM_VCH_WIDTH]] - CHID of the corresponding channel • [hauser[CHID_WIDTH + NUM_VCH_WIDTH-1:0]] - The hard-wired channel identifier. Such as 0 - VCH0, 1 - VCH1, etc.

Connection

AHB5 Interconnect or AHB5-to-SRAM

Table A-6: CONTEXT AHB5 signals

Signal	Direction	Description
haddr_ctxif[31:0]	Output	The byte address of the transfer
hburst_ctxif[2:0]	Output	Indicates how many transfers are in the burst
hmastlock_ctxif	Output	Indicates that the current transfer is part of a locked sequence. The DMA does not initiate locked transfers, therefore it ties this signal to LO
hprot_ctxif[6:0]	Output	Protection control signal, which provides information about the access type.
hsize_ctxif[2:0]	Output	Indicates the size of the transfer
hnonsec_ctxif	Output	Indicates whether the transfer is Secure or Non-secure. Not present when SECEXT_PRESENT is set to 0.
htrans_ctxif[1:0]	Output	Indicates the transfer type.
hwdata_ctxif[DATA_WIDTH-1:0]	Output	Transfers data from the Manager to the Subordinate during write operations.
hwstrb_ctxif[DATA_WIDTH/8-1:0]	Output	Write strobes. The DMA doesn't initiate sparse write operations, therefore it drives hwstrb to all-HIGH during the data phase of write operations.
hwrite_ctxif	Output	Indicates the transfer direction.
hrdata_ctxif[DATA_WIDTH-1:0]	Input	Read data bus.
hready_ctxi	Input	When HIGH, the hready indicates to the Manager and all Subordinates, that the previous transfer is complete.
hresp_ctxif	Input	Transfer response.

Trigger signals

The number of trigger ports are set base on NUM_TRIGGER_IN and NUM_TRIGGER_OUT. The selected trigger port type is not present on the entity when these are set to 0.

Connection

Trigger capable peripherals

Table A-7: Trigger signals

Signal	Direction	Description
trig_in_<CH>_src_req	Input	Trigger input request for source trigger port of virtual channel CH. Asynchronous input

Signal	Direction	Description
trig_in_<CH>_src_req_type[1:0]	Input	Trigger input request type for source trigger port of virtual channel CH. Asynchronous input
trig_in_<CH>_src_ack	Output	Trigger input acknowledge for source trigger port of virtual channel CH.
trig_in_<CH>_src_ack_type[1:0]	Output	Trigger input acknowledge type for source trigger port of virtual channel CH.
trig_in_<CH>_dst_req	Input	Trigger input request for destination trigger port of virtual channel CH. Asynchronous input
trig_in_<CH>_dst_req_type[1:0]	Input	Trigger input request type for destination trigger port of virtual channel CH. Asynchronous input
trig_in_<CH>_dst_ack	Output	Trigger input acknowledge for destination trigger port of virtual channel CH
trig_in_<CH>_dst_ack_type[1:0]	Output	Trigger input acknowledge type for destination trigger port of virtual channel CH.
trig_out_<CH>_req	Output	Trigger output request for virtual channel CH.
trig_out_<CH>_ack	Input	Trigger output acknowledge for virtual channel CH. Asynchronous input

IRQ signals

Connection

Interrupt controller

Table A-8: Interrupt request signals

Signal	Direction	Description
irq_channel[NUM_VCH-1:0]	Output	Individual interrupt signal for every channel. Interrupt sources are defined in the register map
irq_comb_nonsec	Output	Interrupt request for DMA unit.
irq_comb_sec	Output	Interrupt request for secure DMA unit.
irq_sec_viol_err	Output	Interrupt request for secure DMA register access violation.

Status and Control signals

Connection

System control

Table A-9: Control signals

Signal	Direction	Description
gpo_ch_<N>[GPO_WIDTH-1:0]	Output	General purpose output for channel N. Not present when CH_GPO_MASK bit N is set to 0.
allch_stop_req_nonsec	Input	All channel stop request for Non-secure channels. When asserted, all Non-secure channels that are not in IDLE are stopped. Channels can be enabled by software, but are immediately stopped if this request is asserted.
allch_stop_ack_nonsec	Output	All channel stopped acknowledge for Non-secure channels. When asserted, all Non-secure channels are stopped or inactive. Four-phase handshake pair of allch_stop_req_nonsec.
allch_pause_req_nonsec	Input	All channel pause request for Non-secure channels. When asserted, then all Non-secure channels that are not in IDLE are paused. Channels can be enabled by software, but are immediately paused if this request is asserted. When the request is deasserted, the operation continues.
allch_pause_ack_nonsec	Output	All channel pause acknowledge for Non-secure channels. When asserted, all Non-secure channels are paused or inactive. Four-phase handshake pair of allch_pause_req_nonsec.
allch_stop_req_sec	Input	All channel stop request for Secure channels. When asserted, all Secure channels that are not in IDLE are stopped. Channels can be enabled by software, but are immediately stopped if this request is asserted. Not present when SECEXT_PRESENT is set to 0.

Signal	Direction	Description
allch_stop_ack_sec	Output	All channel stopped acknowledge for Secure channels. When asserted, all Secure channels are stopped or inactive. Four-phase handshake pair of allch_stop_req_sec. Not present when SECEXT_PRESENT is set to 0.
allch_pause_req_sec	Input	All channel pause request for Secure channels. When asserted, then all Secure channels that are not in IDLE are paused. Channels can be enabled by software, but are immediately paused if this request is asserted. When the request is deasserted, the operation continues. Not present when SECEXT_PRESENT is set to 0.
allch_pause_ack_sec	Output	All channel pause acknowledge for Secure channels. When asserted, all Secure channels are paused or inactive. Four-phase handshake pair of allch_pause_req_sec. Not present when SECEXT_PRESENT is set to 0.

Table A-10: Status signals

Signal	Direction	Description
ch_enabled[NUM_CHANNELS-1:0]	Output	Status indication per channel to show when a channel is active. When SECEXT_PRESENT is set to 1, then channels must be separated based on the ch_nonsec signals.
ch_err[NUM_CHANNELS-1:0]	Output	Status indication per channel to show when a channel has encountered an error. When SECEXT_PRESENT is set to 1, then channels must be separated based on the ch_nonsec signals.
ch_stopped[NUM_CHANNELS-1:0]	Output	Status indication per channel to show when the channel is stopped. When SECEXT_PRESENT is set to 1, then channels must be separated based on the ch_nonsec signals.
ch_paused[NUM_CHANNELS-1:0]	Output	Status indication per channel to show when the channel is paused. When SECEXT_PRESENT is set to 1, then channels must be separated based on the ch_nonsec signals.
ch_priv[NUM_CHANNELS-1:0]	Output	Status indication per channel to show the privilege setting of the channel. When SECEXT_PRESENT is set to 1, then channels must be separated based on the ch_nonsec signals.
ch_nonsec[NUM_CHANNELS-1:0]	Output	Status indication per channel to show the security setting of the channel. Not present when SECEXT_PRESENT is set to 0.

Cross Trigger Interface

Connection

CoreSight debug

Table A-11: Cross Trigger Interface signals

Signal	Direction	Description
halt_req	Input	Level input to pause all operations and go into a halted state for both Secure and Non-secure channels. Compatible with the Cross Trigger Interface.
restart_req	Input	Pulse input to resume the operation from the halted state. Compatible with the Cross Trigger Interface.
halted	Output	Pulse indication that the DMAC reached the halted state. Compatible with the Cross Trigger Interface.

Configuration and boot signals

Connection

Static configuration values

Table A-12: Configuration signals

Signal	Direction	Description
boot_en	Input	Enables the channel 0 to load a first command after reset from the specified input address set by boot_addr. Sampled once after reset when power and clock have become stable
boot_addr[31:2]	Input	Word aligned address for the boot process. When SECEXT_PRESENT is set to 1, then the boot_addr needs to point to a secure memory region because channel 0 boots as secure. Sampled once after reset when power and clock have become stable and boot_en is HIGH.
boot_priv	Input	Privilege status setting for channel 0. Sampled once after reset when power and clock have become stable and boot_en is HIGH
boot_memattr[7:0]	Input	Boot address memory attributes. Has the same encoding as the LINKMEMATTRHI and LINKMEMATTRLO registers combined. Sampled once after reset when power and clock have become stable and boot_en is HIGH
boot_shareattr[1:0]	Input	Boot address memory shareability attributes. Has the same encoding as the LINKSHAREATTR register. Sampled once after reset when power and clock have become stable and boot_en is HIGH.
initial_cntxbase[31:12]	Input	4-KB aligned context area base pointer. When SECEXT_PRESENT is set to 1, then it will be sampled to SEC_CNTXBASE, otherwise to NSEC_CNTXBASE register. Sampled once after reset when power and clock have become stable.
initial_cntxmem_clr	Input	Enables context memory initialization after reset. Sampled once after reset when power and clock have become stable

Proprietary Notice

This document is protected by copyright and other related rights and the use 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 Limited ("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 the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. 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, any patents, copyrights, trade secrets, trademarks, or other rights.

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.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted 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.

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 this document shall prevail.

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

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Product revision status

The r0p0 identifier indicates the revision status of the product described in this manual, where:

r*x* Identifies the major revision of the product.
p*y* Identifies the minor revision or modification status of the product.

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0000-02	20 September 2024	Non-Confidential	First EAC release
0000-01	20 February 2024	Confidential	Initial release

Change history

The Change history tables describe the technical changes between released issues of this document in reverse order. Issue numbers match the revision history in Document release information on page 174.

Table 2: Issue 0000-01

Change	Location
First release	-

Table 3: Differences between issue 0000-01 and issue 0000-02

Change	Location
Format of the document Front matter and Appendix A changed to match new Arm standards	-
Minor changes to the documentation	Entire document
Description of iBEP User Guide added	Documentation
Context Memory section and subsections added	Context Memory
Context Memory area size section added	Context Memory area size
“Programming considerations” section renamed to “About burst generation on the Data AHB5 interface”, and moved	Configuring burst generation on the Data AHB5 interface
Multiple changes to Error handling topic	Error handling
Getting started section added	Getting started
hauser_dmaif signal, in Data AHB signal table updated	Signal descriptions
Burst values in table updated	Configuring burst generation on the Data AHB5 interface

Conventions

The following subsections describe conventions used in Arm documents.

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: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

Convention	Use
<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	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

We recommend the following. If you do not follow these recommendations your system might not work.

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.

This information is important and needs your attention.

This information might help you perform a task in an easier, better, or faster way.

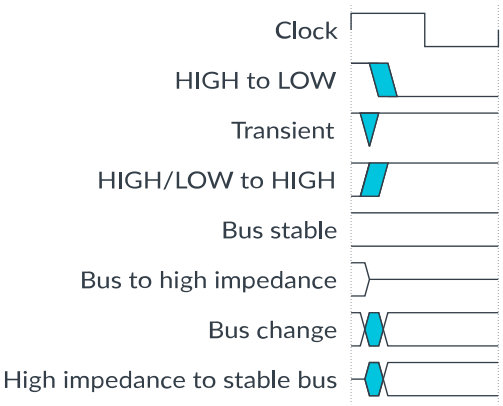
This information reminds you of something important relating to the current content.

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, n denotes an active-LOW signal.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® CoreLink™ DMA-250 Controller Configuration and Integration Manual	108000	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® 5 AHB Protocol Specification	IHI 0033	Non-Confidential
AMBA® APB Protocol Specification	IHI 0024	Non-Confidential
AMBA® Low Power Interface Specification	IHI 0068	Non-Confidential
TrustZone® technology for Armv8-M Architecture	100690	Non-Confidential