



Arm[®] CoreLink[™] MHU-320AE Message Handling Unit

Revision r0p1

Technical Reference Manual

Non-Confidential

Issue 03

Copyright © 2023–2024 Arm Limited (or its affiliates). 107612_0001_03_en
All rights reserved.



Arm® CoreLink™ MHU-320AE Message Handling Unit Technical Reference Manual

This document is Non-Confidential.

Copyright © 2023–2024 Arm Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted [Arm's Proprietary Notice](#) found at the end of this document.

This document (107612_0001_03_en) was issued on 2024-05-17. There might be a later issue at <http://developer.arm.com/documentation/107612>

The product revision is r0p1.

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 MHU-320AE Message Handling Unit.

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. Overview of MHU-320AE.....	6
1.1 Features of MHU-320AE.....	7
1.2 Compliance.....	8
1.3 Product documentation.....	9
2. Functional description of MHU-320AE.....	10
2.1 MHU-320AE structure.....	10
2.2 MHU Sender.....	12
2.3 MHU Receiver.....	13
2.4 MHU bridge.....	14
2.5 MHU-320AE channels.....	14
2.5.1 Doorbell channels.....	15
2.5.2 Fast channels.....	16
2.5.3 FIFO channels.....	18
3. Interfaces.....	23
3.1 MHU Sender interfaces.....	23
3.2 MHU Receiver interfaces.....	26
4. Operations of MHU-320AE.....	29
4.1 RAMs and ECC.....	29
4.1.1 RAM error simulation.....	29
4.1.2 RAM scrub.....	30
4.2 Power management.....	30
4.3 Reliability, Accessibility, and Serviceability.....	31
4.3.1 Error record classification.....	32
4.3.2 Error recovery and fault handling interrupts.....	32
4.3.3 Error handling records.....	33
4.3.4 Bus errors.....	38
5. Programmers model for MHU-320AE.....	39
5.1 Register map pages.....	40
5.2 Discovery.....	40

5.2.1 Discovery flow.....	41
5.3 MHU FMU register summary.....	42
5.3.1 FMU_ERR<n>FR, Error Record <n> Feature Register, n = 0 - 5.....	43
5.3.2 FMU_ERR<n>CTLR, Error Record <n> Control Register, n = 0 - 5.....	45
5.3.3 FMU_ERR<n>STATUS, Error Record <n> Status Register, n = 0 - 5.....	46
5.3.4 FMU_ERRGSR, Error Group Status Register.....	49
5.3.5 FMU_ERRIIDR, Implementation Identification Register.....	50
5.3.6 FMU_SMEN, Safety Mechanism Enable Register.....	51
5.3.7 FMU_SMERR, Safety Mechanism Error Register.....	53
5.3.8 FMU_SMCR, Safety Mechanism Set Criticality Register.....	54
5.3.9 FMU_SMWR, Safety Mechanism Page Write Register.....	55
5.3.10 FMU_SMWDATA, Safety Mechanism Page Write Data Register.....	56
5.3.11 FMU_SMRD, Safety Mechanism Page Read Register.....	57
5.3.12 FMU_SMRDATA, Safety Mechanism Page Read Data Register.....	59
5.3.13 FMU_STATUS, FMU Status Register.....	60
5.3.14 FMU_KEY, FMU Key Register.....	62
5.3.15 FMU_TIMEOUT, FMU Timeout Duration Register.....	63
5.3.16 FMU_ERRUPDATE, FMU Error Update Register.....	64
5.3.17 FMU_FCTLR, FMU Feature Control Register.....	65
5.3.18 FMU_ERRDEVARCH, Device Architecture Register.....	66
5.3.19 FMU_ERRDEVID, Device Configuration Register.....	67
5.3.20 FMU_PIDR4, MHU FMU Peripheral ID 4 Register.....	68
5.3.21 FMU_PIDR5, MHU FMU Peripheral ID 5 Register.....	69
5.3.22 FMU_PIDR6, MHU FMU Peripheral ID 6 Register.....	69
5.3.23 FMU_PIDR7, MHU FMU Peripheral ID 7 Register.....	70
5.3.24 FMU_PIDR0, MHU FMU Peripheral ID 0 Register.....	71
5.3.25 FMU_PIDR1, MHU FMU Peripheral ID 1 Register.....	72
5.3.26 FMU_PIDR2, MHU FMU Peripheral ID 2 Register.....	73
5.3.27 FMU_PIDR3, MHU FMU Peripheral ID 3 Register.....	74
5.3.28 FMU_CIDR0, MHU FMU Component ID 0 Register.....	75
5.3.29 FMU_CIDR1, MHU FMU Component ID 1 Register.....	76
5.3.30 FMU_CIDR2, MHU FMU Component ID 2 Register.....	77
5.3.31 FMU_CIDR3, MHU FMU Component ID 3 Register.....	77
5.4 MHU Sender registers.....	78
5.4.1 MHU Sender Security Control register summary.....	79
5.4.2 MHU Sender Postbox register summary.....	105

5.4.3 MHU Sender RAS register summary.....	165
5.5 MHU Receiver registers.....	195
5.5.1 MHU Receiver Security Control register summary.....	195
5.5.2 MHU Receiver Mailbox register summary.....	222
5.5.3 MHU Receiver RAS register summary.....	299
6. Functional safety features of MHU-320AE.....	330
6.1 Protection mechanisms of MHU-320AE.....	330
6.2 Fault management unit.....	331
6.2.1 FMU APB5 interface.....	332
6.2.2 Error signaling from a MHU-320AE block to the FMU.....	333
6.2.3 Error signaling by the FMU.....	333
6.2.4 Error record format.....	333
6.2.5 FMU reset.....	334
6.2.6 Protection mechanism IDs.....	334
6.3 Error recovery procedures.....	337
6.3.1 Enable or disable a protection mechanism.....	339
6.3.2 Enable or disable both error signals on a block.....	340
6.3.3 Discover the active errors on the block.....	340
6.3.4 Inject an error in a protection mechanism.....	341
6.4 Ping mechanisms.....	341
6.5 Software interaction.....	342
6.6 FuSa programmer's view.....	344
6.7 FuSa I/O.....	344
6.8 External error inputs.....	345
6.9 Resets.....	345
6.9.1 Cold reset sequence.....	345
6.10 RAM protection.....	346
Proprietary notice.....	347
Product and document information.....	349
Product status.....	349
Revision history.....	349
Conventions.....	351
Useful resources.....	354

1. Overview of MHU-320AE

Arm® CoreLink™ MHU-320AE Message Handling Unit facilitates interrupt-based communication between processing elements executing independent software stacks.

MHU-320AE consists of MHU Sender and MHU Receiver blocks. Each of these blocks has a programming interface for enabling software access to the MHU, as well as interrupt wires to notify the system of events related to the transfer of messages by the MHU.

MHU-320AE implements the [Message Handling Unit Architecture version 3.0](#) with configurable support for every extension in the architecture, including:

- Doorbell Extension (DBE)
- First-In First-Out (FIFO) Extension (FE)
- Fast Channel Extension (FCE)
- TrustZone® Extension (TZE)
- Realm Management Extension (RME)

For more information about the architecture extensions, see the [Message Handling Unit Architecture version 3.0](#).



MHU-320AE enables unidirectional communication between two processing elements. Separate MHU instances are expected to be used to enable full duplex communication.

Functional safety support

You can configure Functional Safety (FuSa) support so that MHU-320AE can be used in safety critical systems and applications. When FuSa support is enabled, MHU-320AE uses the following protection mechanisms:

- Lock-step of core MHU logic blocks
- RAM protection
- AMBA® AXI5-Stream or ACE5-Lite interconnect protection
- AMBA external interface protection
- Q-Channel protection
- Systematic fault watchdog
- Clock and reset duplication
- Fault Management Unit (FMU)

For more information about the Functional Safety (FuSa) configuration parameter, see the *Arm® CoreLink™ MHU-320AE Message Handling Unit Configuration and Integration Manual*.

1.1 Features of MHU-320AE

MHU-320AE provides services for message transfer through multiple channels and channel types, interrupt generation, registers, and programming.

MHU-320AE supports error detection and correction features. MHU-320AE provides flexibility during configuration allowing individual features to be added or removed, allowing the MHU to be tailored to specific use cases.

Message transfer

MHU-320AE provides the following features for transferring messages:

- Support for the following channel types:
 - Up to 128 doorbell channels. Each channel holds 32 individual single-bit flags.
 - Up to 1024 32-bit fast channels or 512 64-bit fast channels
 - Up to 64 First In First Out (FIFO) channels holding up to 1024 bytes of data for each channel
- Support for different system topologies:
 - Choice of a monolithic, single domain MHU instantiation, or distributed instantiations
 - Separate clock and power Q-Channel interfaces for each domain for independent low-power control
 - Use of AXI5-Stream or ACE5-Lite as the communications interface between MHU Sender and MHU Receiver blocks

Interrupt generation

MHU-320AE provides support for the following interrupt types:

- Postbox and mailbox combined interrupts
- Channel transfer interrupts
- Channel transfer acknowledge interrupts
- Fast channel group interrupts
- FIFO tidemark and flush interrupts

Registers and programming

MHU-320AE provides the following programming features:

- Use of APB5 or ACE5-Lite as the programming interface separately in the MHU Sender and MHU Receiver
- TrustZone or Realm support for providing access security

Error correction and containment

MHU-320AE provides the following error correction features:

- Arm® RASv1.1-architecture compliant error reporting for:
 - Software access errors
 - Error Correcting Code (ECC) errors
- Error recovery and fault handling interrupts.

For more information about RASv1.1 architecture, see the RAS System Architecture chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Hardware error detection

Hardware error features are available only if Functional Safety (FuSa) support is enabled.

MHU-320AE provides the following hardware error detection features:

- Lock-step of core MHU logic blocks
- AMBA parity on ACE5-Lite, AXI5-Stream, APB5, and Q-Channel interfaces
- Duplicated reset and clock signals with consistency detection
- Duplicated interrupt outputs
- End-to-end Cyclic Redundancy Check (CRC) protection over AXI5-Stream and ACE5-Lite
- Fault Monitoring Unit (FMU) for error logging that can be placed together with the MHU Sender or MHU Receiver block or both

1.2 Compliance

MHU-320AE complies with various AMBA specifications and standards.

This product is compliant with:

- [Message Handling Unit Architecture version 3.0](#)
- [AMBA® AXI-Stream Protocol Specification](#)
- [AMBA® AXI Protocol Specification](#)
- [AMBA® APB Protocol Specification](#)
- [AMBA® Low Power Interface Specification](#)
- Arm® RAS-architecture compliant error reporting. For more information about RASv1.1 architecture, see the RAS System Architecture chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#)

This document must be read with these specifications. Information from these specifications is not reproduced in this document. [Useful resources](#) on page 354 provides the document numbers of these specifications.

1.3 Product documentation

Each MHU-320AE document is aimed at a particular audience and is associated with specific tasks in the design flow.

These documents do not reproduce information that is available in the Arm architecture and protocol specifications. For architecture and protocol information that relates to MHU-320AE, see the Useful resources section.

The MHU-320AE documentation comprises:

Technical Reference Manual

The Technical Reference Manual (TRM) describes the functionality and the effects of functional options on the behavior of MHU-320AE. This document is useful at all stages of the product design flow.

The choices that are made in the design flow can mean that some behaviors that the TRM describes are not relevant. If you are programming MHU-320AE, then contact:

- The implementer to determine:
 - The build configuration of the implementation
 - The integration, if any, that was performed before implementing MHU-320AE
- The integrator to determine the pin configuration of the device that you use

Configuration and Integration Manual

The Configuration and Integration Manual (CIM) contains:

- Descriptions of MHU-320AE features
- Design-time configuration options
- Reset-time configuration options
- Available build configuration options and related considerations
- Instructions for configuring the RTL with the build configuration options
- Instructions for running test vectors
- Sign-off processes for the configured design
- Considerations when integrating MHU-320AE into your system

The Arm product deliverables include reference scripts and information about using these scripts to implement your design. The reference methodology flows that Arm supplies are example reference implementations only. For EDA tool support, contact your EDA tool vendor.

The CIM is a Confidential document that is only available to licensees of MHU-320AE.

2. Functional description of MHU-320AE

MHU-320AE consists of two major components, the MHU Sender (MHUS) and the MHU Receiver (MHUR).

Each component has a dedicated programming interface and interrupt outputs for software interaction, as well as independent Q-Channel interfaces for low-power control. Both components communicate with each other through an internal MHU Communications interface. MHU-320AE also includes an optional asynchronous bridge.

2.1 MHU-320AE structure

You can configure MHU-320AE to have the MHU Sender and MHU Receiver located in the same domain or in separate domains.

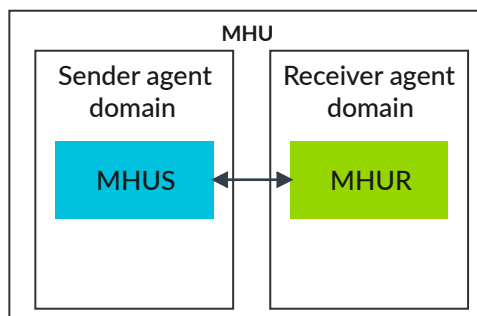
Communication between MHU Sender and MHU Receiver is unidirectional. To enable duplex communication, you must use separate instances of MHU-320AE.

The `STRUCTURE_TYPE` configuration parameter controls the hierarchy and structure of MHU components in given configurations:

mono

Monolithic MHU-320AE configuration with both MHU Sender and MHU Receiver in the same domain.

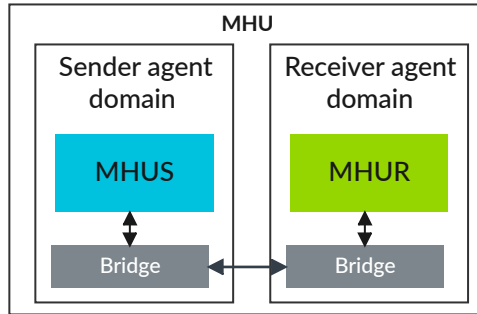
Figure 2-1: Monolithic MHU configuration



full

Single MHU-320AE with MHU Sender and MHU Receiver being in separate domains connected by a bridge.

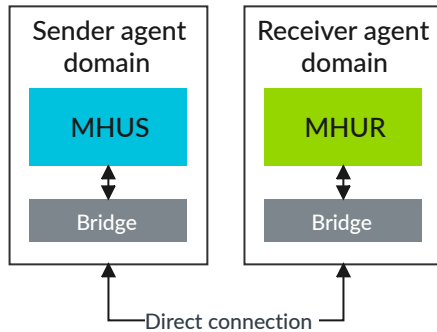
Figure 2-2: Full MHU configuration



bridge

Separate top-level MHU-320AE instances for MHU Sender and MHU Receiver with both being in separate domains. Includes MHU asynchronous bridge for domain crossing. The MHU Sender domain must be connected directly to the MHU Receiver domain.

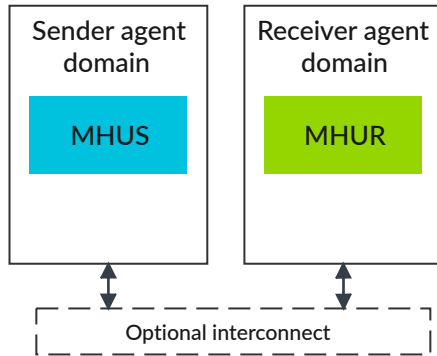
Figure 2-3: Bridge MHU configuration



domain

Separate top-level MHU-320AE instances for MHU Sender and MHU Receiver without including an asynchronous bridge between them. You can use any protocol compliant AXI5-Stream or ACE5-Lite interconnect for routing messages from MHU Sender to MHU Receiver. MHU-320AE does not provide an interconnect.

Figure 2-4: Domain MHU configuration



For more information about configuration parameters, see the *Arm® CoreLink™ MHU-320AE Message Handling Unit Configuration and Integration Manual*.

2.2 MHU Sender

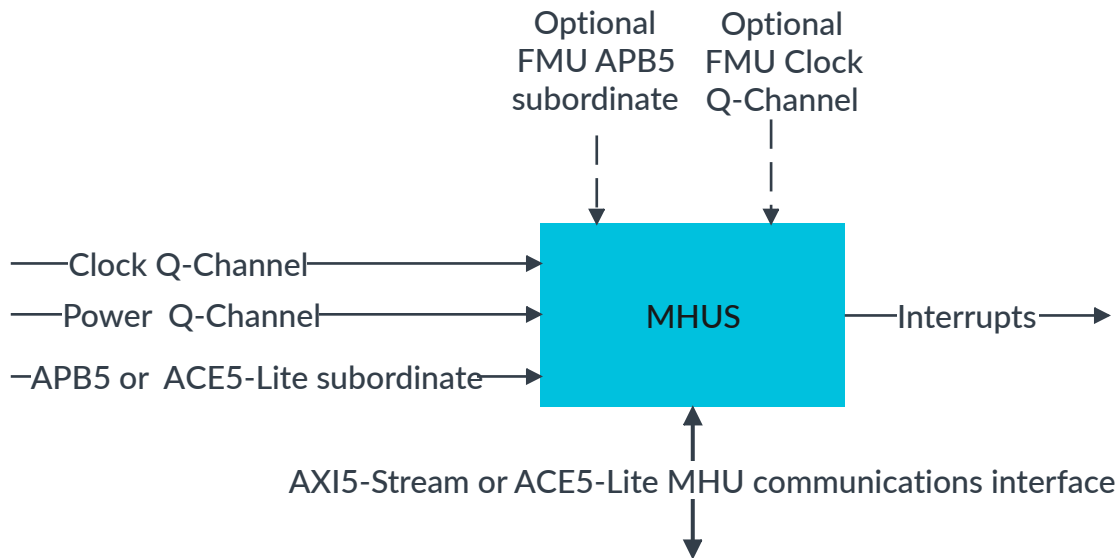
The MHU Sender maps onto the MHU Sender (MHUS) of the [Message Handling Unit Architecture version 3.0](#). It receives transfers from a Sender Agent and forwards them to a Receiver Agent. It also generates interrupts when the receiver notifies it that a sent transfer has been acknowledged.

To send a transfer to the MHU Receiver, the MHU Sender uses a collection of channels and control registers, referred to as a Postbox (PBX).

In configurations with `FUSA_PRESENT == 1` and `FMU_LOCATION != receiver`, the MHU Sender also contains an FMU block for reporting detected hardware faults.

The following figure shows the MHU Sender and its main interfaces.

Figure 2-5: MHU Sender (MHUS)



2.3 MHU Receiver

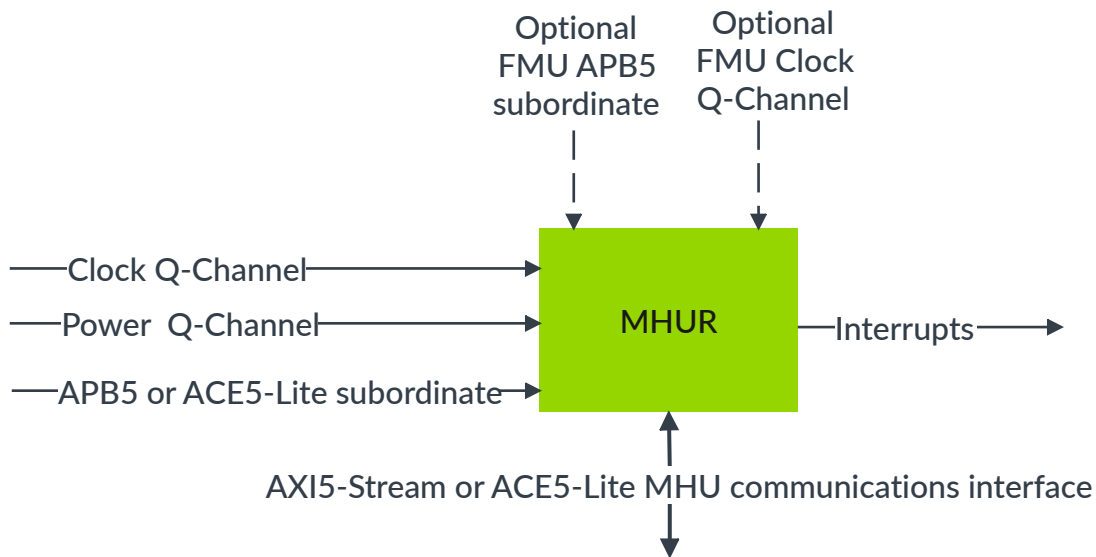
The MHU Receiver component maps onto the MHU Receiver (MHUR) of the [Message Handling Unit Architecture version 3.0](#). It generates interrupts when receiving any transfers from the MHU Sender and delivers them to the Receiver Agent.

The MHU Receiver uses a collection of channels and control registers, referred to as a Mailbox (MBX).

In configurations with `FUSA_PRESENT == 1` and `FMU_LOCATION != sender`, the MHU Receiver also contains an FMU block for reporting detected hardware faults.

The following figure shows the MHU Receiver and its main interfaces.

Figure 2-6: MHU Receiver (MHUR)



2.4 MHU bridge

MHU-320AE includes an optional asynchronous domain bridge, which is used when the MHU structure is bridge or full.

2.5 MHU-320AE channels

MHU-320AE supports the use of the following channels for message transfers.

- [Doorbell channels](#)
- [Fast channels](#)
- [FIFO channels](#)

You must configure MHU-320AE to use at least one type of channels in your design. For more information about MHU-320AE configuration parameters, see the *Arm® CoreLink™ MHU-320AE Message Handling Unit Configuration and Integration Manual*.

2.5.1 Doorbell channels

You can configure MHU-320AE to have up to 128 doorbell channels.

Each doorbell channel holds 32 individual single-bit flags that can be used as event indicators to the MHU Receiver. The MHU Sender can send multiple transfers at once using a single doorbell channel, provided each transfer uses a different flag within the channel.

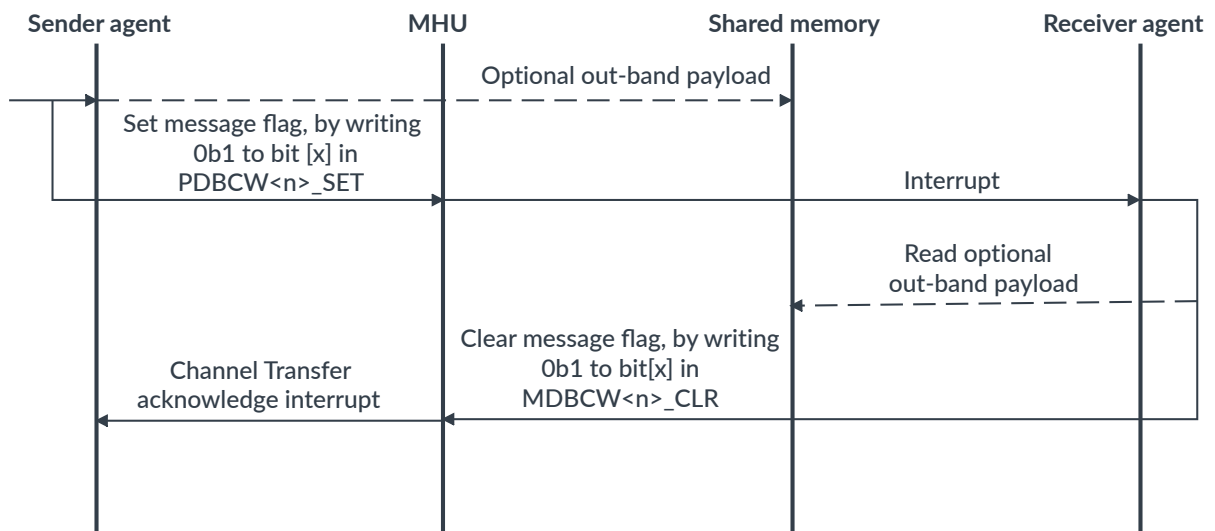
A doorbell channel transfer includes the following steps:

1. When the Sender agent writes 0b1 to any bit of the PDBCW<n>_SET register, a doorbell channel transfer is initiated.
2. A doorbell channel transfer interrupt is sent to the MHU Receiver, unless all set flags are masked.
3. The Receiver agent can then acknowledge the receipt of a doorbell channel transfer by writing the corresponding bits of MDBCW<n>_CLR to 0b1.
4. A doorbell transfer acknowledge interrupt is sent to the MHU Sender, if doorbell channel transfer acknowledge interrupts are enabled.

The Receiver can use the MDBCW<n>_MSK_SET and MDBCW<n>_CLR registers to prevent specific doorbell channel flags from generating transfer events.

The following sequence diagram shows the doorbell transport protocol.

Figure 2-7: Doorbell transfer protocol



For more information, see Doorbell Extension in the [Message Handling Unit Architecture version 3.0](#).

Related information

- [PDBCW<n>_SET](#)

- [MDBCW<n>_CLR](#)
- [MDBCW<n>_MSK_SET](#)

2.5.1.1 Doorbell channel error recovery procedure

If an uncorrectable doorbell channel error occurs, then the data being transferred in this channel is lost together with the channel configuration information.

The following error record registers contain status information on uncorrectable doorbell channel errors:

- [SRAS_ERR1STATUS](#) - Sender view errors in MHU Receiver for all channel types.
- [RRAS_ERR3STATUS](#) - Receiver doorbell channel errors.

Any uncorrectable doorbell channel error reported in the MHU Receiver results in the same error being subsequently reported in the MHU Sender. For more information, see [Reliability, Accessibility, and Serviceability](#).

For all uncorrectable doorbell channel errors, the channel transfer data and any corrupted configuration information are set back to their respective reset values, after which the channel is ready to use without performing any additional recovery steps. Software must only ensure that the channel configuration is reprogrammed as necessary and that the MHU Sender and MHU Receiver have been aligned with what data is expected to be sent next.

Related information

[SRAS_ERR<n>STATUS](#) on page 171

[RRAS_ERR<n>STATUS](#) on page 304

2.5.2 Fast channels

You can configure the MHU-320AE to have up to 1024 fast channels.

Each fast channel can store 32 or 64 bits of transfer data, depending on your configuration. A fast channel holds the last value written to it, similar to a memory.

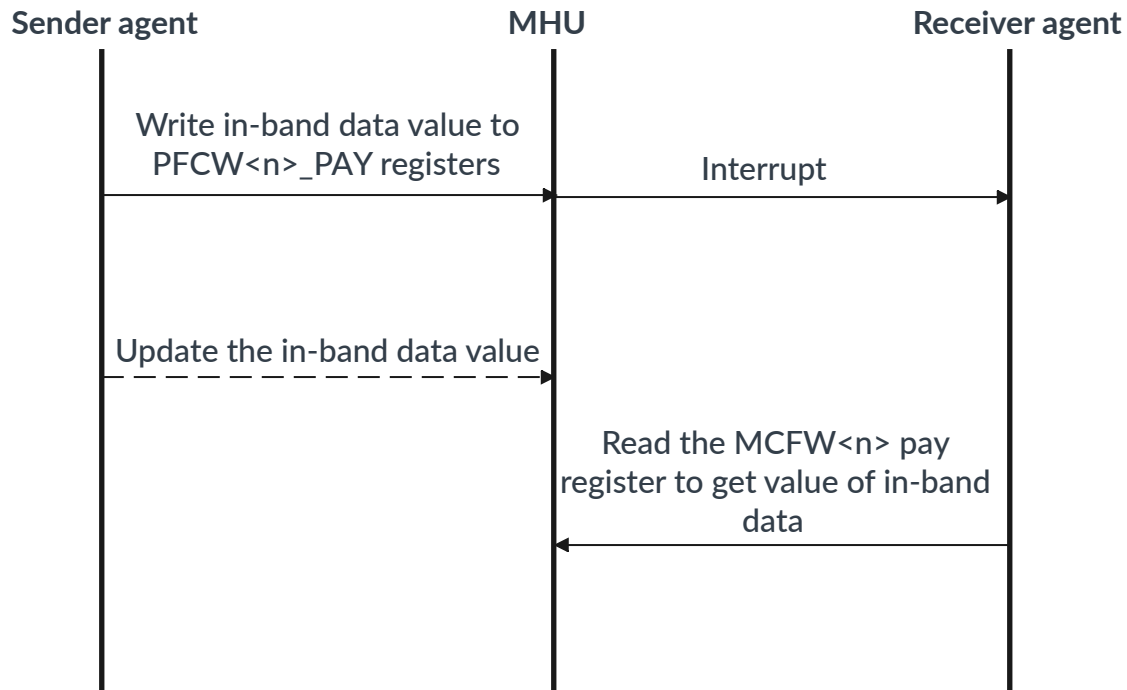
1. When the Sender agent writes to the relevant [PFCW<n>_PAY](#) register, a fast channel transfer is initiated.
2. MHU-320AE forwards the written data to the Receiver agent, where it can be read through the relevant [MFCW<n>_PAY](#) register.
3. The Receiver agent can signal a fast channel transfer interrupt, if enabled in the [MBX_FCG_INT_EN](#) register.

There are no transfer acknowledge events associated with fast channels, therefore the MHU Sender does not have an architectural way of knowing when a transfer has been read by the MHU Receiver.

To help manage a large number of fast channels, you can have up to 32 fast channel groups.

The following sequence diagram shows the last-value transport protocol, which is used for fast channels.

Figure 2-8: Fast channel transport protocol



For more information, see Fast Channel Extension in the [Message Handling Unit Architecture version 3.0](#).

Related information

- [PFCW<n>_PAY32 register](#)
- [PFCW<n>_PAY64 register](#)
- [MFCW<n>_PAY32 register](#)
- [MFCW<n>_PAY64 register](#)
- [MBX_FCG_INT_EN](#)

2.5.2.1 Fast channel error recovery procedure

If an uncorrectable fast channel error occurs, then the data being transferred in this channel is lost together with the channel configuration information.

The following error record registers contain status information on uncorrectable fast channel errors:

- SRAS_ERR1STATUS register - Sender view errors in MHU Receiver for all channel types.
- RRAS_ERR5STATUS register - Receiver fast channel errors.

Any uncorrectable fast channel error reported in the MHU Receiver also results in the same error being subsequently reported in the MHU Sender. For more information, see [Reliability, Accessibility, and Serviceability](#).

For all uncorrectable fast channel errors, the channel transfer data and any corrupted configuration information are set back to their respective reset values, after which the channel is ready to use without performing any additional recovery steps. Software must only ensure that the channel configuration is reprogrammed as necessary and that the MHU Sender and MHU Receiver have been aligned with what data is expected to be sent next.

Related information

[SRAS_ERR<n>STATUS](#) on page 171

[RRAS_ERR<n>STATUS](#) on page 304

2.5.3 FIFO channels

MHU-320AE can be configured to have up to 64 FIFO channels. Each FIFO channel can store up to 1024 bytes of transfer data.

Software can associate individual transfer bytes as being the Start of Transfer (SOT) and the End of Transfer (EOT) and whether the receipt of it generates a transfer acknowledgement event. Therefore if there is free space, a FIFO channel can store multiple transfers at any given time that can vary in length.

Before sending a FIFO channel transfer, the MHU Sender can configure the Transfer Delineation Mode (TDM) by writing to the PFFCW<n>_CTRL.TDM register bit that defines how transfer flags are updated in the MHU-320AE:

Software flag

Flags are fully managed by software.

Partial flag

Flags are partially updated by the MHU with software being able to mark the start and end of transfers when necessary.

Auto flag

Flags are fully managed by the MHU.

The Sender agent can then additionally configure specific transfer flags or a requirement for acknowledgement by writing to PFFCW<n>_FLG. A FIFO channel transfer is initiated when the MHU Sender writes to the PFFCW<n>_PAY register where the amount of data being pushed is determined by the access size. MHU-320AE does not support 8-bit or 16-bit accesses to FIFO channels.

When a FIFO transfer is received with both EOT and ACK flags set, it can signal a FIFO channel transfer interrupt in the MHU Receiver, if enabled. The corresponding data can be read by the Receiver agent through the MFFCW<n>_PAY register. The programmed value of the MFFCW<n>_CTRL.RA_EN register bit determines if a read to the FIFO channel also pops the data:

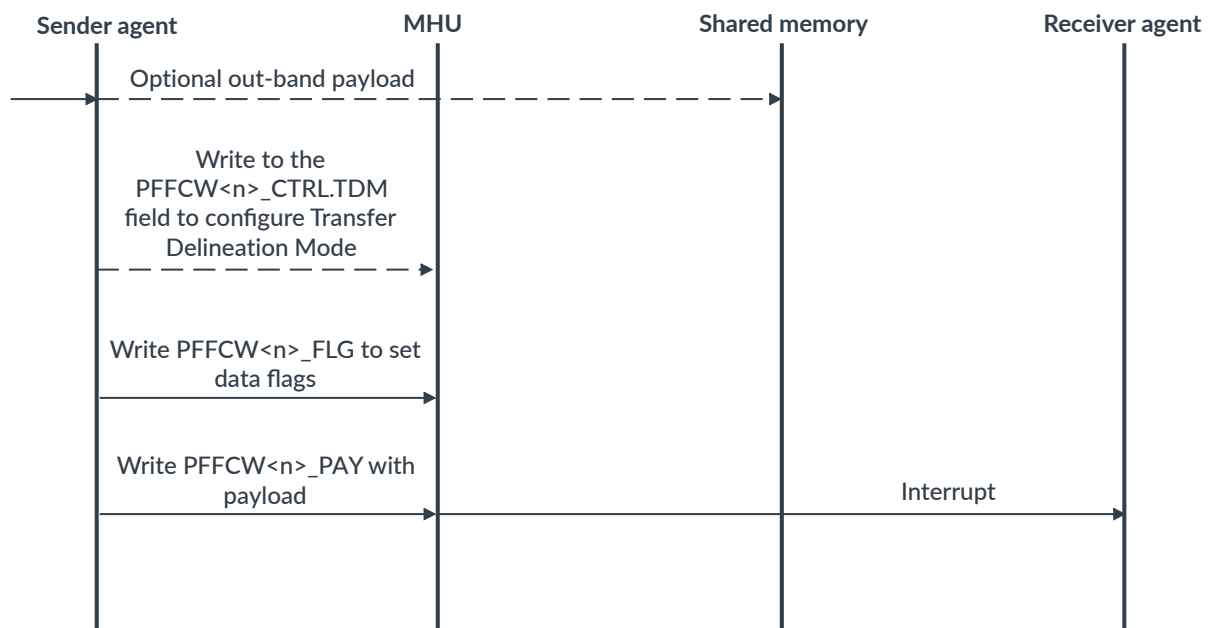
- RA_EN == 1: Read of FIFO data also pops it from FIFO channel.
- RA_EN == 0: Read of FIFO data does not pop it from FIFO channel. Software additionally writes to the MFFCW<n>_FIFO_POP register to pop the data from the FIFO channel.

After the Receiver agent reads a FIFO channel transfer, software can read the associated flags through the MFFCW<n>_FLG register. If one or more bytes are popped from the FIFO channel with both the ACK and EOT flags set, the flags can signal a FIFO channel transfer acknowledgement interrupt in the Sender agent, if enabled. As multiple transfers may have been acknowledged by the Receiver agent when the Receiver agent processes this event, it reads the PFFCW<n>_ACK_CNT register to determine how many transfers have been acknowledged. The action of reading this register clears it. A FIFO channel transfer acknowledge event is only generated when the ACK_CNT field becomes nonzero after being zero before.

Additionally the Sender agent and Receiver agent can also use the FIFO low and high tide events to know when the FIFO channel fill level reaches predefined thresholds on pushing or popping, for example, to send transfers that exceed the depth of a FIFO channel.

The following sequence diagrams shows a simple FIFO transport protocol.

Figure 2-9: FIFO transport protocol write



After the write in the FIFO transport protocol is completed, the sequence depends on whether read-acknowledge is enabled.

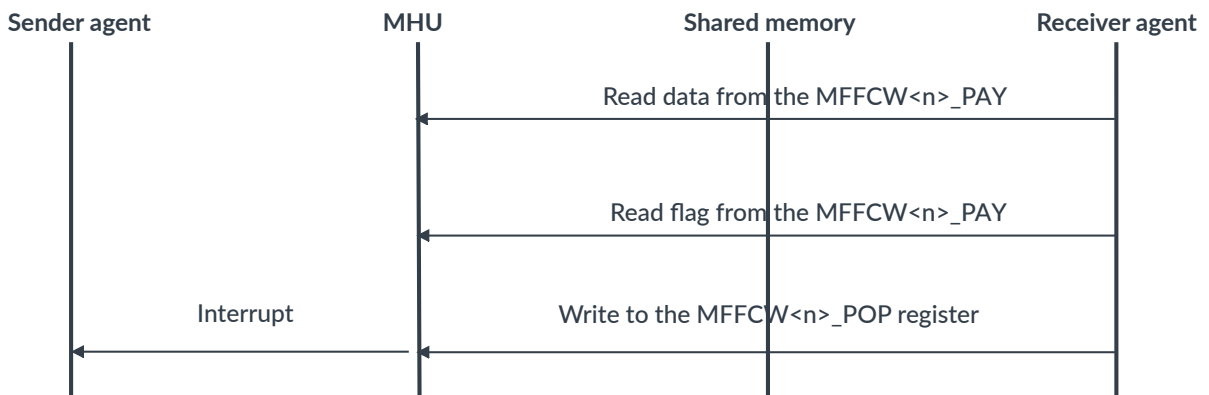
The following sequence occurs if read-acknowledge is enabled.

Figure 2-10: FIFO transport protocol read if read acknowledge is enabled



The following sequence occurs if read-acknowledge is not enabled.

Figure 2-11: FIFO transport protocol read if read acknowledge is disabled



For more information, see FIFO Extension in the [Message Handling Unit Architecture version 3.0](#).

Related information

- [PFFCW<n>_PAY32 register](#)
- [PFFCW<n>_PAY64 register](#)
- [PFFCW<n>_FL32 register](#)
- [PFFCW<n>_FL64 register](#)
- [PFFCW<n>_CTRL register](#)
- [PFFCW<n>_ACK_CNT register](#)
- [MFFCW<n>_PAY32 register](#)
- [MFFCW<n>_PAY64 register](#)
- [MFFCW<n>_FLG32 register](#)

- [MFFCW<n>_FLG64 register](#)
- [MFFCW<n>_CTRL register](#)
- [MFFCW<n>_FIFO_POP register](#)

2.5.3.1 FIFO channel flush

FIFO channel flush is a mechanism that returns a FIFO channel to a clean state following an unexpected event that causes the channel state to get corrupted.

FIFO channel flush can be triggered from either the MHU Sender or MHU Receiver and clears all of the transfer data and flags associated with that channel. To perform a FIFO flush, software must perform the following steps:

1. Set the control bit to 0b1.
2. Wait for the corresponding status bit to become 0b1.
3. Set the control bit to 0b0.
4. Poll for the corresponding status bit to become 0b0.

The control and status registers involved depend on whether the FIFO channel flush is triggered from the MHU Sender or MHU Receiver:

- The MHU Sender writes to PFFCW<n>_CTRL.FF control register and polls the PFFCW<n>_ST.FF status register.
- The MHU Receiver writes to the MFFCW<n>_CTRL.FF control register and polls the MFFCW<n>_ST.FF status register.

For more information, see FIFO flush in the [Message Handling Unit Architecture version 3.0](#).

Related information

[PFFCW<n>_CTRL register](#) on page 152

[PFFCW<n>_ST register](#) on page 155

[MFFCW<n>_CTRL register](#) on page 279

[MFFCW<n>_ST register](#) on page 282

2.5.3.2 FIFO channel error recovery procedure

If an uncorrectable FIFO channel error occurs, then the data being transferred in this channel may have been corrupted and a recovery sequence needs to be performed by software to bring the FIFO channel back to a known state.

The following error record registers contain status information on uncorrectable FIFO channel errors:

- SRAS_ERR1STATUS register - Sender view errors in MHU Receiver for all channel types
- SRAS_ERR3STATUS register - Sender FIFO channel errors

- RRAS_ERR1STATUS register - Receiver view of MHU Sender FIFO channel errors
- RRAS_ERR7STATUS register - Receiver FIFO channel configuration errors
- RRAS_ERR9STATUS register - Receiver FIFO channel data errors

Any uncorrectable FIFO channel error reported in the MHU Receiver also results in the same error being subsequently reported in the MHU Sender.

For more information on error handling procedures, see [Reliability, Accessibility, and Serviceability](#).

For uncorrectable errors, software is required to perform the following recovery sequence:

1. Read the error record to determine if an uncorrectable error has occurred and record the corrupted FIFO channel number.
2. Clear the error record to enable tracking of any future errors.
3. If an error has occurred in the MHU Receiver, the MHU Sender must wait for the corresponding error to be reported in SRAS_ERR1STATUS.
4. Ensure the FIFO channel is empty and any remaining transfers have been popped for FIFO channel by the MHU Receiver.
5. Read PFFCW<n>_ACK_CNT for FIFO channel to determine the number of previously acknowledged transfers, and clear stored ACK_CNT.
6. Reprogram the FIFO channel states, for example the interrupt enables, as required.
7. Align the MHU Sender and the MHU Receiver with what data is expected to be sent next.

Related information

[SRAS_ERR<n>STATUS](#) on page 171

[RRAS_ERR<n>STATUS](#) on page 304

3. Interfaces

MHU-320AE contains multiple interfaces for both the MHU Sender and the MHU Receiver. The presence and properties of an interface depend on the configuration you use.

For more information about the MHU-320AE signals, see the *Arm® CoreLink™ MHU-320AE Message Handling Unit Configuration and Integration Manual*.

3.1 MHU Sender interfaces

The MHU Sender interfaces depend on the configuration used in the design. For more information about the MHU Sender configuration parameters, see the *Arm® CoreLink™ MHU-320AE Message Handling Unit Configuration and Integration Manual*.

MHU Sender APB5 subordinate interface

The MHU Sender APB5 subordinate interface provides access to the register map of the MHU Sender and is only present in configurations where `SENDER/REG_IF_TYPE == apb`.

For information on how MHU-320AE drives the PSLVERR response signal, see [Bus errors](#).

MHU Sender ACE5-Lite subordinate interface

The MHU Sender ACE5-Lite subordinate interface provides access to the register map of the MHU Sender and is only present in configurations where `SENDER/REG_IF_TYPE == acelite`.

The following table shows the acceptance capabilities of the MHU Sender ACE5-Lite subordinate interface.

Table 3-1: MHU Sender ACE5-Lite subordinate interface acceptance capabilities

Attribute	Capability
Combined acceptance capability	2
Read acceptance capability	1
Read data reorder depth	1
Write acceptance capability	1

The following table lists the ACE5-Lite properties for the MHU Sender subordinate interface.

Table 3-2: MHU Sender ACE5-Lite subordinate interface properties

AMBA property	Value	ACE5-Lite issue
Atomic_Transactions	False	F
Barrier_Transactions	False	F
Busy_Support	False	J
Cache_Stash_Transactions	False	F
Check_Type	Odd_Parity_Byte_All when <code>REG_PROTECTION_TYPE == parity</code> , False otherwise	F
CMO_On_Read	False	G

AMBA property	Value	ACE5-Lite issue
CMO_On_Write	False	G
Coherency_Connection_Signals	False	F
DeAllocation_Transactions	False	F
DVM_v8	False	E
DVM_v8.1	False	F
DVM_v8.4	False	H
DVM_v9.2	False	J
Exclusive_Accesses	False	H
InvalidateHint_Transaction	False	J
Loopback_Signals	True	F
Max_Transaction_Bytes	4096	H
MPAM_Support	False	G
MTE_Support	False	H
NSAccess_Identifiers	False	F
Ordered_Write_Observation	True	E
Persist_CMO	False	F
Poison	False	F
Prefetch_Transaction	False	H
QoS_Accept	False	F
Read_Data_Chunking	False	G
Read_Interleaving_Disabled	True	G
Regular_Transaction_Only	False	H
RME_Support	True when REG_SECURITY_TYPE == rme	J
Shareable_Transactions	True	H
Trace_Signals	True	F
Unique_ID_Support	True	G
Untranslated_Transactions	False	F
Wakeup_Signals	True	F
Write_Plus_CMO	False	H
WriteDeferrable_Transaction	False	J
WriteZero_Transaction	False	H

For information on how MHU-320AE drives the RRESP and BRESP response signals, see [Bus errors](#).

MHU Sender FMU APB5 subordinate interface

The MHU Sender FMU APB5 subordinate interface provides access to the register map of the MHU Sender FMU and is only present in FuSa configurations where `FMU_LOCATION != receiver`.

MHU Sender interrupts

The MHU Sender optionally provides the following interrupt outputs:

- Postbox combined (always present)
- Error recovery and fault handling (always present)
- Channel transfer acknowledgement
- FIFO flush and tidemark
- Critical and non-critical fault

All MHU Sender interrupts are active-HIGH and level-sensitive.

MHU Sender Q-Channel interfaces

The MHU Sender has the following Q-Channel device interfaces for low power control:

- Clock Q-Channel
- Power Q-Channel - controls MHU Sender entry into non-operational state
- FMU Clock Q-Channel - only present when `FMU_LOCATION != receiver`

For more information about Q-Channel interfaces, see the [AMBA® Low Power Interface Specification](#).



In configurations with `STRUCTURE_TYPE == mono`, the MHU Sender clock and power Q-Channel interfaces get combined with the respective MHU Receiver Q-Channel interfaces.

MHU Sender AXI5-Stream interface

The MHU Sender uses a bidirectional AXI5-Stream interface to communicate to the MHU Receiver when `MSG_IF_TYPE == axit`.



You must not reorder or interleave packets between the MHU Sender and MHU Receiver, regardless of the interconnect being used between these endpoints.

MHU Sender ACE5-Lite communications interfaces

The MHU Sender uses a pair of separate subordinate and manager ACE5-Lite interfaces to communicate to the MHU Receiver when `MSG_IF_TYPE == acelite`.

The properties are the same as those as listed in [Table 3-2: MHU Sender ACE5-Lite subordinate interface properties](#) on page 23, with the exception of Loop and Trace signals on the manager port, which are not supported.

3.2 MHU Receiver interfaces

The MHU Receiver interfaces depend on the configuration used in the design. For more information about the MHU Receiver configuration parameters, see the *Arm® CoreLink™ MHU-320AE Message Handling Unit Configuration and Integration Manual*.

MHU Receiver APB5 subordinate interface

The MHU Receiver APB5 subordinate interface provides access to the register map of the MHU Receiver and is only present in configurations where `RECEIVER/REG_IF_TYPE == apb`.

For information on how MHU-320AE drives the PSLVERR response signal, see [Bus errors](#).

MHU Receiver ACE5-Lite subordinate interface

The MHU Receiver ACE5-Lite subordinate interface provides access to the register map of the MHU Receiver and is only present in configurations where `RECEIVER/REG_IF_TYPE == acelite`.

The following table shows the acceptance capabilities of the MHU Receiver ACE5-Lite subordinate interface.

Table 3-3: MHU Receiver ACE5-Lite subordinate interface acceptance capabilities

Attribute	Capability
Combined acceptance capability	4
Read acceptance capability	2
Read data reorder depth	1
Write acceptance capability	2

The following table lists the ACE5-Lite properties for the MHU Receiver subordinate interface.

Table 3-4: MHU Receiver ACE5-Lite subordinate interface properties

AMBA property	Value	ACE5-Lite issue
Atomic_Transactions	False	F
Barrier_Transactions	False	F
Busy_Support	False	J
Cache_Stash_Transactions	False	F
Check_Type	Odd_Parity_Byte_All when <code>REG_PROTECTION_TYPE == parity</code> , False otherwise	F
CMO_On_Read	False	G
CMO_On_Write	False	G
Coherency_Connection_Signals	False	F
DeAllocation_Transactions	False	F
DVM_v8	False	E
DVM_v8.1	False	F
DVM_v8.4	False	H
DVM_v9.2	False	J
Exclusive_Accesses	False	H

AMBA property	Value	ACE5-Lite issue
InvalidateHint_Transaction	False	J
Loopback_Signals	True	F
Max_Transaction_Bytes	4096	H
MPAM_Support	False	G
MTE_Support	False	H
NSAccess_Identifiers	False	F
Ordered_Write_Observation	True	E
Persist_CMO	False	F
Poison	False	F
Prefetch_Transaction	False	H
QoS_Accept	False	F
Read_Data_Chunking	False	G
Read_Interleaving_Disabled	True	G
Regular_Transaction_Only	False	H
RME_Support	True when REG_SECURITY_TYPE == rme	J
Shareable_Transactions	True	H
Trace_Signals	True	F
Unique_ID_Support	True	G
Untranslated_Transactions	False	F
Wakeup_Signals	True	F
Write_Plus_CMO	False	H
WriteDeferrable_Transaction	False	J
WriteZero_Transaction	False	H

For information on how MHU-320AE drives the RRESP and BRESP response signals, see [Bus errors](#).

MHU Receiver FMU APB5 subordinate interface

The MHU Receiver FMU APB5 subordinate interface provides access to the register map of the MHU Receiver FMU and is only present in FuSa configurations where `FMU_LOCATION != sender`.

MHU Receiver interrupts

The MHU Receiver optionally provides the following interrupt outputs:

- Mailbox combined (always present)
- Error recovery and fault handling (always present)
- Channel transfer
- FIFO flush and tidemark
- Critical and non-critical fault

All MHU Receiver interrupts are active-HIGH and level-sensitive.

MHU Receiver Q-Channel interfaces

The MHU Receiver has the following Q-Channel device interfaces for low power control:

- Clock Q-Channel
- Power Q-Channel - controls MHU Receiver entry into non-operational state
- FMU clock Q-Channel - only present when `FMU_LOCATION != sender`

For more information about Q-Channel interfaces, see the [AMBA® Low Power Interface Specification](#).



In configurations with `STRUCTURE_TYPE == mono`, the MHU Receiver clock and power Q-Channel interfaces get combined with the respective MHU Receiver Q-Channel interfaces.

MHU Receiver AXI5-Stream interface

The MHU Receiver uses a bidirectional AXI5-Stream interface to communicate to the MHU Receiver when `MSG_IF_TYPE == axit`.



Irrespective of the interconnect being used between the MHU Receiver and MHU Receiver, packets must not be reordered or interleaved between these endpoints.

MHU Receiver ACE5-Lite communications interfaces

The MHU Receiver uses a pair of separate subordinate and manager ACE5-Lite interfaces to communicate to the sender when `MSG_IF_TYPE == acelite`.

The properties are the same as those as listed in [Table 3-4: MHU Receiver ACE5-Lite subordinate interface properties](#) on page 26, with the exception of Loop and Trace signals on the manager port, which are not supported.

4. Operations of MHU-320AE

The details of the MHU-320AE operation depend on the configuration used in your design.

For the details of steps involved in each transport protocol, see the following sections:

- [Doorbell channels](#)
- [Fast channels](#)
- [FIFO channels](#)

[RAMs and ECC](#) are used to store and protect channel-specific information, including transfer data. Configurable [Reliability, Accessibility, and Serviceability](#) features protect the information stored in the RAMs.

MHU-320AE uses MHU Sender and MHU Receiver Q-Channel interfaces for [Power management](#).

4.1 RAMs and ECC

MHU-320AE can use multiple RAMs to store a range of channel specific information, including transfer data. In typical operation, the RAMs are transparent to software.

Each RAM is protected from errors using an ECC with Single Error Correction and Double Error Detection (SECCDED).

If single or double errors are detected, they are reported in the software visible error records, see [Reliability, Accessibility, and Serviceability](#) for more information.

4.1.1 RAM error simulation

For each RAM present in the configuration, software can use the following registers to simulate a transient ECC single-bit or double-bit error:

- PBX_FIFO_ERRINS register: Sender FIFO channel RAM
- MBX_DB_ERRINS register: Receiver doorbell channel RAM
- MBX_FST_ERRINS register: Receiver fast channel RAM
- MBX_FCFG_ERRINS register: Receiver FIFO channel configuration RAM
- MBX_FDATA_ERRINS register: Receiver FIFO channel data RAM

These registers cause an error to be inserted, to a specified address and location in the associated RAM. The ECC encoder and decoder are checked but the RAM content is not modified. These registers can only be accessed by the most trusted security state present in the configuration.

After software inserts an error, MHU-320AE reports the error in the associated error record, in the same manner as any regular ECC error. However, the software injected error has no effect on the functionality of the MHU, so software can inject errors injection during operation.

If a coincident real error occurs, then the MHU reports the real error instead and triggers the normal containment mechanism for that channel type.

Related information

- [PBX_FIFO_ERRINS register](#)
- [MBX_DB_ERRINS register](#)
- [MBX_FST_ERRINS register](#)
- [MBX_FCFG_ERRINS register](#)
- [MBX_FDATA_ERRINS register](#)

4.1.2 RAM scrub

MHU-320AE can hold channel specific programming and transfer information in RAMs, which is protected by Single Error Correction and Double Error Detection (SECCDED).

However, some RAM contents might be static for a long duration, and there is a potential for errors to accumulate if a particular address is not periodically accessed. To prevent this occurring, software can periodically trigger a low-priority scrub of a RAM, by setting the PBX_FCTRL.SIP and MBX_FCTRL.SIP register bits in the MHU Sender and MHU Receiver respectively. This process triggers a check and if necessary, a write-back of all valid RAM entries. Any errors that are found during a scrub are also reported in the relevant RAS error records.

Related information

- [PBX_FCTRL](#)
- [MBX_FCTRL](#)

4.2 Power management

Each domain present in a MHU-320AE configuration exposes a power Q-Channel interface that allows the system power controller to power down the corresponding domain.

This interface also controls the entry of the MHU into the non-operational architectural state. In configurations where the MHU Sender and MHU Receiver are in separate domains, both domains act independently in terms of power control.

For the MHU domain to accept entry into the non-operational state, all of the following statements must be true:

- No outstanding messages are being sent between the MHU Sender and MHU Receiver.
- No programming accesses are being performed to respective domain.
- All doorbell and FIFO channels present in configuration are idle.
- All fast channels present in configuration are idle, if the domain contains an MHU Receiver.

- FIFO flush operations are not outstanding.

Software can also prevent the MHU Sender or MHU Receiver from entering the non-operational state by setting the PBX_CTRL.OP_REQ or MBX_CTRL.OP_REQ register fields respectively.

For the purposes of entering the non-operational state, a channel is considered non-idle if it has unread or unacknowledged data stored in it. When entering a non-operational state software can disregard non-idle channels by setting the PBX_CTRL.CH_OP_MSK or MBX_CTRL.CH_OP_MSK register fields for the domain that is being powered down.

To ensure that MHU-320AE is isolated from the system, MHU-320AE must enter the non-operational state before reset is asserted.

In configurations where `FUSA_PRESENT == 1`, if the power Q-channel fails to respond due to a hardware fault, a full system reset may be necessary.

For more detailed information about the non-operational state and power control, see the [Message Handling Unit Architecture version 3.0](#).

Related information

- [PBX_CTRL](#)
- [MBX_CTRL](#)

4.3 Reliability, Accessibility, and Serviceability

MHU-320AE uses a range of configurable RAS features for all RAMs, which include Single Error Correction and Double Error Detection (SECCDED), and scrub, software and bus error reporting.

The configuration parameters to enable RAS features for RAMs related to a given channel type are:

- Doorbell channels: `RAS_DB_PRESENT`
- Fast channels: `RAS_FAST_PRESENT`
- FIFO channels: `RAS_FIFO_PRESENT`

The MHU makes all necessary information available to software through Arm RASv1.1 architecture-compliant register space. For more information about RASv1.1 architecture, see the RAS System Architecture chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

4.3.1 Error record classification

The MHU reports errors in Arm RASv1.1 architecture-compliant error records, separately in the MHU Sender and MHU Receiver, which are accessible through the respective subordinate programming interfaces.

- The MHU Sender error records are placed in the SRAS register block in the MHU Sender. For more information, see the [External MHUS register summary](#).
- The MHU Receiver error records are placed in the RRAS register block in the MHU Receiver, see the [External MHUR register summary](#).

The classes of error records are:

- Correctable ECC errors
- Uncorrectable ECC errors
- Software access errors



In configurations where multiple security states are present software can use SRAS_ERRACR and RRAS_ERRACR registers in the MHU Sender and MHU Receiver respectively to control which security states are allowed to perform accesses to MHU error records.

For more information about RASv1.1 architecture, see the RAS System Architecture chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

4.3.2 Error recovery and fault handling interrupts

You can assign a recorded correctable or uncorrectable error to the fault handling interrupt (fault_int signal) by setting the associated ERR<n>CTLR.FI register field. For more information, see either [SRAS_ERR<n>CTLR](#) or [RRAS_ERR<n>CTLR](#).

All correctable ECC errors have error counters, where by default the interrupt would only be signaled when the counter in the associated ERR<n>MISCO register overflows. For more information, see either [SRAS_ERR<n>MISCO](#) or [RRAS_ERR<n>MISCO](#). This counter can be preset to any value by writing to ERR<n>MISCO.Count. For example:

- To fire an interrupt on any correctable error, write 0xFF
- To fire an interrupt on every second correctable error, write 0xFE

Alternatively, the correctable error record can be configured to signal a fault handling interrupt on any recorded error by setting the associated ERR<n>CTRL.CED field.

For uncorrectable error records, fault handling interrupts are generated on every recorded error, if enabled. Uncorrectable error records can also generate an error recovery interrupt (err_int signal), by setting the associated ERR<n>CTLR.UI field. The interrupt fires on every uncorrectable interrupt occurrence, irrespective of the counter value.

4.3.3 Error handling records

Both the MHU Sender and MHU Receiver have several error records. The range of available error handling records depends on the configuration of MHU-320AE.



MHU-320AE follows the recommended RAS extension implementation of the MHUv3.0 architecture with the associated ERR<n>MISCO register recording the following:

- Transfer data lost: bit [16]
- Impact zone: bits [14:13]
- Channel type: bits [12:10]
- Channel number: bits [9:0]

The following tables summarize the MHU-320AE error handling records in the MHU Sender and MHU Receiver. The Type column uses the following terms:

- Correctable Error (CE)
- Uncorrected latent, or restartable error (UEO)
- Uncorrected error, signalled, or recoverable error (UER)

The following table lists the MHU Sender error handling records.

Table 4-1: MHU Sender error handling records

Record	Description	Type	Further information
0	Software error in MHU Sender programming	UEO	MHU Sender and MHU Receiver software error records, Record 0
1	Uncorrectable MHU Receiver errors	UER	Uncorrectable MHU Receiver error record 1 in MHU Sender
2	Correctable MHU Sender FIFO RAM errors	CE	MHU Sender FIFO channel RAM error records 2 and 3
3	Uncorrectable MHU Sender FIFO RAM errors	UER	MHU Sender FIFO channel RAM error records 2 and 3

The following table lists the MHU Receiver error handling records.

Table 4-2: Receiver error handling records

Record	Description	Type	Further information
0	Software error in MHU Receiver programming	UEO	MHU Sender and MHU Receiver software error records, Record 0
1	Uncorrectable MHU Sender errors	UER	Uncorrectable MHU Sender error record 1 in MHU Receiver
2	Correctable MHU Receiver doorbell RAM errors	CE	MHU Receiver doorbell channel RAM error records 2 and 3
3	Uncorrectable MHU Receiver doorbell RAM errors	UER	MHU Receiver doorbell channel RAM error records 2 and 3
4	Correctable MHU Receiver fast channel RAM errors	CE	MHU Receiver fast channel RAM error records 4 and 5
5	Uncorrectable MHU Receiver fast channel RAM errors	UER	MHU Receiver fast channel RAM error records 4 and 5
6	Correctable MHU Receiver FIFO configuration RAM errors	CE	MHU Receiver FIFO channel configuration RAM error records 6 and 7

Record	Description	Type	Further information
7	Uncorrectable MHU Receiver FIFO configuration RAM errors	UER	MHU Receiver FIFO channel configuration RAM error records 6 and 7
8	Correctable MHU Receiver FIFO data RAM errors	CE	MHU Receiver FIFO channel data RAM error records 8 and 9
9	Uncorrectable MHU Receiver FIFO data RAM errors	UER	MHU Receiver FIFO channel data RAM error records 8 and 9

4.3.3.1 MHU Sender and MHU Receiver software error records, Record 0

Software error record 0 contains software errors that are uncorrectable both in the MHU Sender and MHU Receiver.

Record 0 contains software programming errors from a wide range of sources within the MHU-320AE. In general, these errors are contained. For uncorrected errors, the information that is provided gives enough information to enable recovery without significant loss of functionality.

We recommend that record 0 is connected to a high priority interrupt within the system. This connection prevents the record from overflowing if it receives more errors than it is able to process with the possible loss of information that is required for recovery.

The following table describes the syndromes that are recorded in record 0, the reported information, and recovery instructions.

Table 4-3: Software errors, record 0

ERR<n>STATUS.IERR (Syndrome)	ERR<n>STATUS.SERR	ERR<n>MISC1 data description	Recovery and prevention
0x0, SYN_REG_BAD Illegal subordinate access	0xE	AccessRnW, bit [11] AccessSize, bits [10:8] AccessLength, bits [7:0]	Repeat illegal access, with appropriate size and properties. Full access address is given in ERR<n>ADDR.
0x1, SYN_REC_DB_CORRUPTED Data read from doorbell channel that encountered an uncorrectable error	0x6	None	Software has tried to read corrupted data that is stored in MHU Receiver doorbell channel RAM. Check the relevant RAM error record. Full access address is given in ERR<n>ADDR in MHU Receiver.
0x2, SYN_REC_FST_CORRUPTED Data read from fast channel that encountered an uncorrectable error	0x6	None	Software has tried to read corrupted data that is stored in MHU Receiver fast channel RAM. Check the relevant RAM error record. Full access address is given in ERR<n>ADDR in MHU Receiver.
0x3, SYN_FIFO_CORRUPTED Data read from FIFO channel that encountered an uncorrectable error	0x6	None	Software has tried to read corrupted data that is stored in MHU Sender or MHU Receiver FIFO channel RAM. Check the relevant RAM error record. Full access address is given in ERR<n>ADDR.
0x4, SYN_ACE_CC_BAD Illegal ACE5-Lite subordinate access on communications interface	0xE	LenErr, bit [2] StrbErr, bit [1] LastErr, bit [0]	The MHU communications interface received an ACE5-Lite access of an unexpected type, as indicated by the MISC1 information. The communications interface expects a single-beat, 64-bit access.



You can use the SRAS_ERR0CTLR.DAE and RRAS_ERR0CTLR.DAE bits in the MHU Sender and MHU Receiver to disable error reporting for illegal subordinate accesses (SYN_REG_BAD syndrome). For more information, see either [SRAS_ERR<n>CTLR](#) or [RRAS_ERR<n>CTLR](#).

4.3.3.2 Uncorrectable MHU Receiver error record 1 in MHU Sender

Error record 1 in the MHU Sender contains uncorrectable RAM errors that have been observed in the MHU Receiver. You can use this error record to let the MHU Sender software know that a particular channel has been corrupted, even if the corruption is not local, in case the MHU Sender software needs to take action. You can obtain the channel type and number information from SRAS_ERR1MISC0 register.

4.3.3.3 MHU Sender FIFO channel RAM error records 2 and 3

MHU Sender FIFO channel RAM error record 2 contains RAM ECC errors that are correctable. MHU Sender FIFO channel RAM error record 3 contains RAM ECC errors that are uncorrectable.

If a correctable error is detected in the MHU Sender FIFO channel RAM, it is corrected and the error is reported in error record 2.

For information about the error counters and interrupt generation options, see [Error recovery and fault handling interrupts](#).

Correctable errors do not require software to take any action within the MHU. However, software can choose to track error locations in case a RAM row or column can be repaired, and the RAM has repair capability.

The following table lists the [SRAS_ERR<n>MISC1](#) report data for MHU Sender FIFO channel RAM error records 2 and 3.

Table 4-4: Sender FIFO channel RAM errors, records 2 and 3

Record	ERR<n>MISC1 data description
2 = Correctable	Bit location, bits [5:0]
3 = Uncorrectable	None

For more information, see [FIFO channel error recovery procedure](#).

4.3.3.4 Uncorrectable MHU Sender error record 1 in MHU Receiver

Error record 1 in the MHU Receiver contains uncorrectable RAM errors that have been observed in the MHU Sender. The aim of this error record is to let the MHU Receiver software know that a particular channel has been corrupted in case it needs to take action, even if the corruption is not local. The channel type and number information can be obtained from RRAS_ERR1MISC0.

4.3.3.5 MHU Receiver doorbell channel RAM error records 2 and 3

MHU Receiver doorbell channel RAM error record 2 contains RAM ECC errors that are correctable and record 3 contains RAM ECC errors that are uncorrectable.

If a correctable error is detected in the MHU Receiver doorbell channel RAM, it is corrected and the error is reported in error record 2. For information about the error counters and interrupt generation options, see [Error recovery and fault handling interrupts](#).

Correctable errors do not require software to take any action within the MHU. However, software can choose to track error locations if a RAM row or column can be repaired, and the RAM has repair capability.

The following table lists the [RRAS_ERR<>MISC1](#) report data for MHU Receiver doorbell channel RAM error records 2 and 3.

Table 4-5: Receiver doorbell channel RAM errors, records 2 and 3

Record	ERR<n>MISC1 data description
2 = Correctable	Bit location, bits [6:0]
3 = Uncorrectable	None

For more information, see [Doorbell channel error recovery procedure](#).

4.3.3.6 MHU Receiver fast channel RAM error records 4 and 5

MHU Receiver fast channel RAM error record 4 contains RAM ECC errors that are correctable and record 5 contains RAM ECC errors that are uncorrectable.

If a correctable error is detected in the MHU Receiver fast channel RAM, it is corrected and the error is reported in error record 4. For information about the error counters and interrupt generation options, see [Error recovery and fault handling interrupts](#).

Correctable errors do not require software to take any action within the MHU. However, software can choose to track error locations in case a RAM row or column can be repaired, and the RAM has repair capability.

The following table lists the [RRAS_ERR<>MISC1](#) reports data for MHU Receiver fast channel RAM error records 4 and 5.

Table 4-6: Receiver fast channel RAM errors, records 4 and 5

Record	ERR<n>MISC1 data description
4 = Correctable	Bit location, bits [5:0]
5 = Uncorrectable	None

For more information, see [Fast channel error recovery procedure](#).

4.3.3.7 MHU Receiver FIFO channel configuration RAM error records 6 and 7

MHU Receiver FIFO channel configuration RAM error record 6 contains RAM ECC errors that are correctable. Receiver FIFO channel configuration RAM error record 7 contains RAM ECC errors that are uncorrectable.

If a correctable error is detected in the MHU Receiver FIFO channel configuration RAM, it is corrected and the error is reported in error record 6. For information about the error counters and interrupt generation options, see [Error recovery and fault handling interrupts](#).

Correctable errors do not require software to take any action within the MHU. However, software can choose to track error locations in case a RAM row or column can be repaired, and the RAM has repair capability.

The following table lists the [RRAS_ERR<>MISC1](#) report data for MHU Receiver FIFO channel configuration RAM error records 6 and 7.

Table 4-7: Receiver FIFO channel configuration RAM errors, records 6 and 7

Record	ERR<n>MISC1 data description
6 = Correctable	Bit location, bits [6:0]
7 = Uncorrectable	None

For more information, see [FIFO channel error recovery procedure](#).

4.3.3.8 MHU Receiver FIFO channel data RAM error records 8 and 9

MHU Receiver FIFO channel data RAM error record 8 contains RAM ECC errors that are correctable. Receiver FIFO channel data RAM error record 9 contains RAM ECC errors that are uncorrectable.

If a correctable error is detected in the MHU Receiver FIFO channel data RAM, it is corrected and the error is reported in error record 8. For information about the error counters and interrupt generation options, see [Error recovery and fault handling interrupts](#).

Correctable errors do not require software to take any action within the MHU. However, software can choose to track error locations in case a RAM row or column can be repaired, and the RAM has repair capability.

The following table lists the [RRAS_ERR<>MISC1](#) report data for MHU Receiver FIFO channel data RAM error records 8 and 9.

Table 4-8: Receiver FIFO channel data RAM errors, records 8 and 9

Record	ERR<n>MISC1 data description
8 = Correctable	Bit location, bits [n+7:n] Address, bits [n-1:0] where $n = \text{ceil}(\log_2(\text{NUM_FIFO_CHAN} * \text{FIFO_CHAN_DEPTH} / 16))$
9 = Uncorrectable	Address, bits [n-1:0]

The corrupted FIFO channel can then be calculated from the reported address by using the following formula:

$$ID = \text{floor}((16 * \text{address}) / \text{FIFO_CHAN_DEPTH})$$

For more information, see [FIFO channel error recovery procedure](#).

4.3.4 Bus errors

APB5 or ACE5-Lite bus error syndromes such as bad transactions, and corrupted RAM data reads can be made to report an APB5 or ACE-Lite external Subordinate response error (SLVERR).

The [SRAS_ERR<n>CTRL.UE](#) and [RRAS_ERR<n>CTRL.UE](#) register bits can be used to enable or disable this error in the MHU Sender or MHU Receiver for the syndromes shown in the following table.

Table 4-9: Bus error syndromes

Syndrome	Description	Access
SYN_REG_BAD	Programming access is either illegal or unrecognized	Read and write
SYN_REC_DB_CORRUPTED	Data read from doorbell channel RAM is corrupted.	Read
SYN_REC_FST_CORRUPTED	Data read from fast channel RAM is corrupted	Read
SYN_FIFO_CORRUPTED	Data read from FIFO channel RAM is corrupted	Read

5. Programmers model for MHU-320AE

The MHU-320AE consists of a MHU Sender component, a MHU Receiver component, and an optional bridge component. The MHU Sender and MHU Receiver is accessed through memory-mapped registers for configuration, topology, and status information. Depending on the configuration, these are accessed via either APB or ACE-Lite reads and writes.

In configurations with `FUSA_PRESENT == 1`, each FMU present has its own APB programming interface in addition to the MHU Sender and MHU Receiver programming interfaces.

The base address of the configuration registers is not fixed and can be different for any particular system implementation. The offset of each register from the configuration base address is fixed.

When accessing the configuration registers, do not attempt to access reserved or unused address locations. Attempting to access these locations can result in unpredictable behavior. Unless otherwise stated in the accompanying text:

- Do not modify undefined register bits.
- Ignore undefined register bits on reads.
- All register bits are reset to 0 by a system or Cold reset.

Each register has an associated access type. The MHU-320AE registers use the following access type abbreviations:

RW

Read and write

RO

Read only

WO

Write only

RAZ

Read as zero

WI

Write ignored

Some bit positions in registers are described as reserved. These bit positions have the following access types:

- **RAZ/WI** in an RW register
- **RAZ** in an RO register
- **WI** in a WO register

The MHU-320AE registers are accessed using the APB completer or ACE-Lite subordinate interfaces. Accesses to unmapped or reserved registers are **WI** or **RAZ**. Non-secure accesses to Secure registers are **WI** or **RAZ**.

MHU-320AE contains several control registers that enable software to modify the behavior of the product. Usually, programming the control registers immediately impacts the execution of transactions that flow through MHU-320AE.

MHU-320AE provides a mechanism for software to discover the configuration of the product. For more information, see [Discovery](#).

5.1 Register map pages

The MHU-320AE MHU Sender and MHU Receiver address maps contain multiple pages. The number of pages and address aliasing depends on the MHU-320AE configuration.

The MHU-320AE registers are grouped into 64KB blocks, each of which is formed of multiple 4KB pages.

MHU-320AE always retains the following block order:

1. MHU Sender or MHU Receiver Security Control block, if `SENDER/REG_SECURITY_TYPE != none` OR `RECEIVER/REG_SECURITY_TYPE != none` respectively
2. Postbox or Mailbox block
3. MHU Sender or MHU Receiver RAS block

For example if the MHU-320AE configuration has MHU Sender Security Control and MHU Receiver Security Control register blocks present in the configuration, the following block offsets are used for the register blocks (same as the programmers model presented in this document):

Table 5-1: MHU blocks with security blocks present

Block offset	Block
0x00000	MHU Sender or MHU Receiver Security Control
0x10000	Postbox or Mailbox
0x20000	MHU Sender or MHU Receiver RAS

Alternatively, the following block offsets are used if both security blocks are not present in the configuration:

Table 5-2: MHU blocks without security blocks present

Block offset	Block
0x00000	Postbox or Mailbox
0x10000	MHU Sender or MHU Receiver RAS

5.2 Discovery

Discovery is an algorithm that software can use to determine the structure of the MHU-320AE configuration as the system boots. Therefore, software can determine the structure of the

MHU-320AE domains, components, and subfeatures without previous knowledge of the configuration.

To build the discovery tree, the discovery process starts at the base address of the configuration space. We recommend that the operating system is provided with pointers to the start of the MHU Sender and MHU Receiver memory maps. Then discovery uses pointer values to determine the number and type of each component, their attributes, and the location of the configuration registers. Software can use this information to access these registers for configuration purposes. For more information, see [Discovery flow](#).

5.2.1 Discovery flow

To verify that the pages relate to MHU-320AE registers, software can check these pointers against the discovery registers, which start at offset 0x0FD0 for each MHU page. These registers allow discovery of the MHU-320AE version as well as information whether the page contains MHU or FMU registers.

To discover the page type, software can:

1. Read from 0x0FE0 to determine the PIDR0.PART_0 value.
2. Read from 0x0FE4 to determine the PIDR1.PART_1 value.
3. Concatenate PART_1 (4 bits) and PART_0 (8 bits), to discover the 12-bit part number, PART_1 || PART_0. A value of:
 - 0x0F7 indicates that this page contains MHU registers.
 - 0x49C indicates that this page contains Sender FMU registers.
 - 0x49D indicates that this page contains Receiver MHU registers.

The type of MHU page can be further determined by:

1. If a read from 0x0FBC returns non-zero data, this page contains MHU SRAS or RRAS registers.
2. If a read from 0x0FBC returns zero, read from 0x000 to determine the MHU block identifier:
 - 0x0 indicates that this page contains Postbox registers
 - 0x1 indicates that this page contains Mailbox registers
 - 0x2 indicates that this page contains Sender Security Control registers
 - 0x3 indicates that this page contains Receiver Security Control registers

When this information is known, software can obtain additional information from registers that are specific to each page, such as architecture version and the supported architectural features.

For more information on discovery and feature registers, see the [Message Handling Unit Architecture version 3.0](#).

5.3 MHU FMU register summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MHU FMU Register Block registers.

For more information on registers listed in the table, click on the link associated with the register name.

Some registers do not have a listed reset value, as they are either write-only or this depends on the particular block or protection mechanism being accessed.

Table 5-3: MHU FMU Register Block register summary

Offset	Name	Type	Reset	Width	Description
0x000 + (64 * n)	FMU_ERR<n>FR	RO	See individual bit resets.	64-bit	Error Record <n> Feature Register
0x008 + (64 * n)	FMU_ERR<n>CTLR	RW	See individual bit resets.	64-bit	Error Record <n> Control Register
0x010 + (64 * n)	FMU_ERR<n>STATUS	RW	See individual bit resets.	64-bit	Error Record <n> Status Register
0xE00	FMU_ERRGSR	RO	See individual bit resets.	64-bit	Error Group Status Register
0xE10	FMU_ERRIIDR	RO	See individual bit resets.	32-bit	Implementation Identification Register
0xF00	FMU_SMEN	WO	See individual bit resets.	32-bit	Safety Mechanism Enable Register
0xF04	FMU_SMERR	WO	See individual bit resets.	32-bit	Safety Mechanism Error Register
0xF08	FMU_SMCR	WO	See individual bit resets.	32-bit	Safety Mechanism Set Criticality Register
0xF0C	FMU_SMWR	WO	See individual bit resets.	32-bit	Safety Mechanism Page Write Register
0xF10	FMU_SMWDATA	RW	See individual bit resets.	32-bit	Safety Mechanism Page Write Data Register
0xF14	FMU_SMRD	WO	See individual bit resets.	32-bit	Safety Mechanism Page Read Register
0xF18	FMU_SMRDATA	RO	See individual bit resets.	32-bit	Safety Mechanism Page Read Data Register
0xF1C	FMU_STATUS	RO	See individual bit resets.	32-bit	FMU Status Register
0xF20	FMU_KEY	RW	See individual bit resets.	32-bit	FMU Key Register
0xF24	FMU_TIMEOUT	RW	See individual bit resets.	32-bit	FMU Timeout Duration Register
0xF28	FMU_ERRUPDATE	WO	See individual bit resets.	32-bit	FMU Error Update Register
0xF2C	FMU_FCTLR	RW	See individual bit resets.	32-bit	FMU Feature Control Register
0xFBC	FMU_ERRDEVARCH	RO	See individual bit resets.	32-bit	Device Architecture Register
0xFC8	FMU_ERRDEVID	RO	See individual bit resets.	32-bit	Device Configuration Register
0xFD0	FMU_PIDR4	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 4 Register
0xFD4	FMU_PIDR5	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 5 Register
0xFD8	FMU_PIDR6	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 6 Register
0xFDC	FMU_PIDR7	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 7 Register
0xFE0	FMU_PIDR0	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 0 Register
0xFE4	FMU_PIDR1	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 1 Register
0xFE8	FMU_PIDR2	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 2 Register
0xFEC	FMU_PIDR3	RO	See individual bit resets.	32-bit	MHU FMU Peripheral ID 3 Register
0xFF0	FMU_CIDR0	RO	See individual bit resets.	32-bit	MHU FMU Component ID 0 Register
0xFF4	FMU_CIDR1	RO	See individual bit resets.	32-bit	MHU FMU Component ID 1 Register
0xFF8	FMU_CIDR2	RO	See individual bit resets.	32-bit	MHU FMU Component ID 2 Register

Offset	Name	Type	Reset	Width	Description
0xFFC	FMU_CIDR3	RO	See individual bit resets.	32-bit	MHU FMU Component ID 3 Register

5.3.1 FMU_ERR<n>FR, Error Record <n> Feature Register, n = 0 - 5

Defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHU FMU Register Block

Register offset

0x000 + (64 * n)

Bit descriptions

Figure 5-1: MHU_FMU_REGISTER_BLOCK_FMU_ERR<n>FR bit assignments

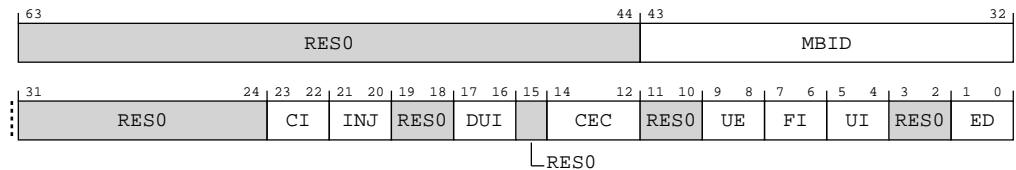


Table 5-4: FMU_ERR<n>FR bit descriptions

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	RES0
[43:32]	MBID	Maximum BLKID. The maximum BLKID for the error record block .	0x0
[31:24]	RES0	Reserved	RES0
[23:22]	CI	<p>Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node.</p> <p>0b00 Does not support the critical error interrupt. FMU_ERR<n>CTLR.CI is RES0.</p> <p>0b10 Critical error interrupt is supported and controllable using FMU_ERR<n>CTLR.CI.</p>	xx

Bits	Name	Description	Reset
[21:20]	INJ	Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node. 0b00 Does not support the Common Fault Injection Model Extension.	0b00
[19:18]	RES0	Reserved	RES0
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node. 0b00 Does not support the enabling and disabling of error recovery interrupts on deferred errors. FMU_ERR<n>CTLR.DUI is RES0 .	0b00
[15]	RES0	Reserved	RES0
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in FMU_ERR<n>MISC0. 0b000 Does not implement the standard format Corrected error counter model.	0b000
[11:10]	RES0	Reserved	RES0
[9:8]	UE	In-band error response (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node. 0b00 Does not support in-band error response. FMU_ERR<n>CTLR.UE is RES0 .	0b00
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node. 0b00 Does not support the fault handling interrupt. FMU_ERR<n>CTLR.FI is RES0 .	0b00
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node. 0b00 Does not support the error handling interrupt. FMU_ERR<n>CTLR.UI is RES0 . 0b10 Error handling interrupt is supported and controllable using FMU_ERR<n>CTLR.UI.	xx
[3:2]	RES0	Reserved	RES0
[1:0]	ED	Error reporting and logging. Indicates error record <n> is the first record owned the node, and whether the node implements the controls for enabling and disabling error reporting and logging. 0b00 Error record not implemented. Error reporting and logging is not enabled for error record. 0b10 Error reporting and logging is controllable using FMU_ERR<n>CTLR.ED.	xx

Accessibility

Table 5-5: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0x000 + (64 * n)	FMU_ERR<n>FR	None

5.3.2 FMU_ERR<n>CTLR, Error Record <n> Control Register, n = 0 - 5

The error control register contains enable bits for the node that writes to this record.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHU FMU Register Block

Register offset

0x008 + (64 * n)

Bit descriptions

Figure 5-2: MHU_FMU_REGISTER_BLOCK_FMU_ERR<n>CTLR bit assignments

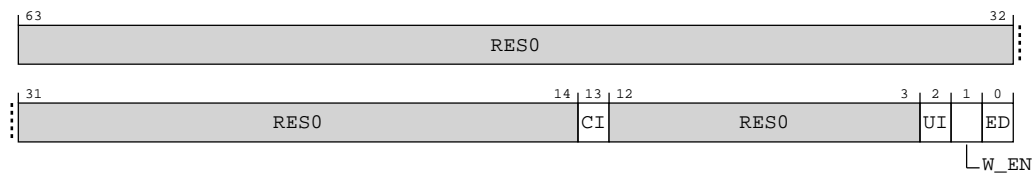


Table 5-6: FMU_ERR<n>CTLR bit descriptions

Bits	Name	Description	Reset
[63:14]	RES0	Reserved	RES0
[13]	CI	Critical error interrupt enable. When enabled, the critical error interrupt is generated for a critical error condition. 0b0 Critical error interrupt not generated for critical errors. Critical errors are treated as Uncontained errors. 0b1 Critical error interrupt generated for critical errors.	0b0
[12:3]	RES0	Reserved	RES0
[2]	UI	Uncorrected error recovery interrupt enable. 0b0 Error recovery interrupt disabled. 0b1 Error recovery interrupt enabled.	0b0

Bits	Name	Description	Reset
[1]	W_EN	Provides the ability to disable error input wires for this record in the situation where a fault causes an error wire to be permanently asserted. This disables the error wire such that it has no software visible effect. 0b0 Error wire inputs disabled for this record. 0b1 Error wire inputs enabled for this record (reset value).	x
[0]	ED	Error reporting and logging enable. When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. 0b0 Error reporting disabled. 0b1 Error reporting enabled.	x

Accessibility

Table 5-7: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0x008 + (64 * n)	FMU_ERR<n>CTLR	None

5.3.3 FMU_ERR<n>STATUS, Error Record <n> Status Register, n = 0 - 5

Contains status information for error record <n>.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHU FMU Register Block

Register offset

0x010 + (64 * n)

Bit descriptions

Figure 5-3: MHU_FMU_REGISTER_BLOCK_FMU_ERR<n>STATUS bit assignments

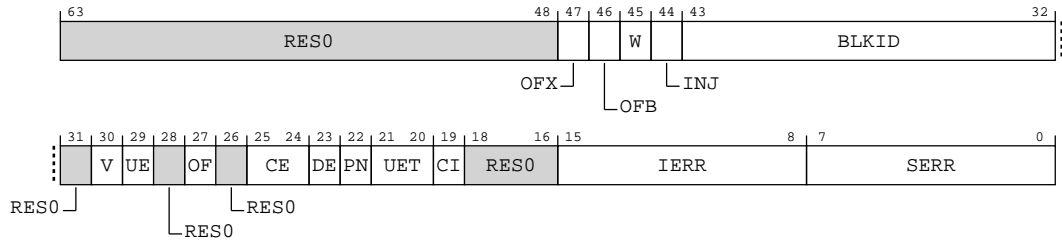


Table 5-8: FMU_ERR<n>STATUS bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47]	OFX	Different BLKID overflowed to the one being reported in BLKID. Cleared by starting ERRUPDATE for this record. 0b0 Overflow with different BLKID not detected. 0b1 Overflow with different BLKID detected.	0b0
[46]	OFB	Reported BLKID has overflowed for a different PROTID to the one being reported in IERR. Cleared by successful clear of V bit. 0b0 Overflow with different PROTID not detected. 0b1 Overflow with different PROTID detected.	0b0
[45]	W	Determines whether error wire is asserted for error record. 0b0 Error wire has not been asserted. 0b1 Error wire has been asserted.	0b0
[44]	INJ	Indicates that the reported error was inserted via FMU_SMERR. This field is not valid if FMU_ERR<n>STATUS.V is set to 0. 0b0 Reported error has not been inserted via FMU_SMERR. 0b1 Reported error has been inserted via FMU_SMERR.	0b0
[43:32]	BLKID	Valid only when FMU_ERR<n>STATUS.V==1.This field indicates the ID of the block which is reporting an error. When BLKID is not known (i.e. when only the error wire is currently received or when software clears V), this field becomes 0.	0x000
[31]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[30]	V	Status Register Valid. 0b0 FMU_ERR<n>STATUS not valid. 0b1 FMU_ERR<n>STATUS valid. At least one error has been recorded.	0b0
[29]	UE	Uncorrected Error. 0b0 No errors have been detected, or all detected errors have been either corrected or deferred. 0b1 At least one detected error was not corrected and not deferred.	0b0
[28]	RES0	Reserved	RES0
[27]	OF	Overflow. Indicates that multiple errors have been detected. 0b0 No error syndrome for an Uncorrected error has been discarded and error counter has not overflowed. 0b1 At least one error syndrome has been discarded or, if an error counter is implemented, it might have overflowed.	0b0
[26]	RES0	Reserved	RES0
[25:24]	CE	Corrected Error. 0b00 No errors were corrected.	0b00
[23]	DE	Deferred Error. 0b0 No errors were deferred.	0b0
[22]	PN	Poison. 0b0 Error not related to a poison value.	0b0
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. 0b11 Uncorrected error, Signaled or Recoverable error (UER).	0b00
[19]	CI	Critical Error. Indicates whether a critical error condition has been recorded. 0b0 No critical error condition. 0b1 Critical error condition.	0b0
[18:16]	RES0	Reserved	RES0
[15:8]	IERR	Contains the PROTID which indicates the protection mechanism reporting the error. If FMU_ERR<n>STATUS.V is set to 0, this field is not valid and reads 0. When V=1 but the protection ID is not known this is set to 0.	0x00
[7:0]	SERR	Architecturally-defined primary error code. See the RAS System Architecture chapter of the Arm® Architecture Reference Manual for A-profile architecture for more information on this field.	0x01

Accessibility

Table 5-9: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0x010 + (64 * n)	FMU_ERR<n>STATUS	None

5.3.4 FMU_ERRGSR, Error Group Status Register

Shows the status for the records in the group.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHU FMU Register Block

Register offset

0xE00

Bit descriptions

Figure 5-4: MHU_FMU_REGISTER_BLOCK_FMU_ERRGSR bit assignments

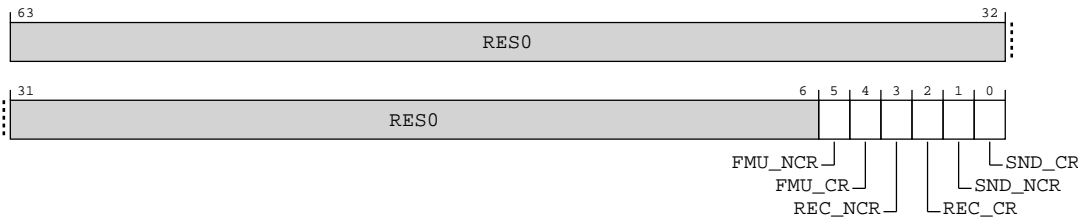


Table 5-10: FMU_ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:6]	RES0	Reserved	RES0
[5]	FMU_NCR	Indicates the status of the FMU non-critical error record. 0b0 The error record is not reporting any errors. 0b1 The error record is reporting one or more errors.	0b0

Bits	Name	Description	Reset
[4]	FMU_CR	Indicates the status of the FMU critical error record. 0b0 The error record is not reporting any errors. 0b1 The error record is reporting one or more errors.	0b0
[3]	REC_NCR	Indicates the status of the Receiver non-critical error record. 0b0 The error record is not reporting any errors. 0b1 The error record is reporting one or more errors.	0b0
[2]	REC_CR	Indicates the status of the Receiver critical error record. 0b0 The error record is not reporting any errors. 0b1 The error record is reporting one or more errors.	0b0
[1]	SND_NCR	Indicates the status of the Sender non-critical error record. 0b0 The error record is not reporting any errors. 0b1 The error record is reporting one or more errors.	0b0
[0]	SND_CR	Indicates the status of the Sender critical error record. 0b0 The error record is not reporting any errors. 0b1 The error record is reporting one or more errors.	0b0

Accessibility

Table 5-11: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xE00	FMU_ERRGSR	None

5.3.5 FMU_ERRIIDR, Implementation Identification Register

Defines the implementer of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xE10

Bit descriptions

Figure 5-5: MHU_FMU_REGISTER_BLOCK_FMU_ERRIIDR bit assignments

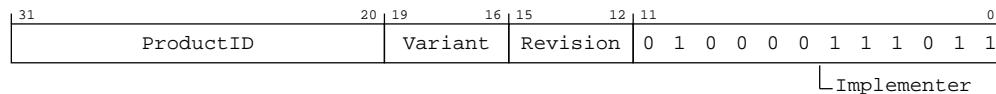


Table 5-12: FMU_ERRIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component.	The reset value depends on whether the FMU is located in the MHU Sender or MHU Receiver. <ul style="list-style-type: none"> 0x49C - MHU Sender FMU 0x49D - MHU Receiver FMU
[19:16]	Variant	Component major revision. This field distinguishes product variants or major revisions of the product.	The reset value depends on the product version used. <ul style="list-style-type: none"> 0x0 - r0
[15:12]	Revision	Component minor revision. This field distinguishes minor revisions of the product.	The reset value depends on the product version used. <ul style="list-style-type: none"> 0x0 - p0 0x1 - p1
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the RAS component. For an Arm implementation, this field has the value 0x43B. 0b010000111011	0x43B

Accessibility

Table 5-13: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xE10	FMU_ERRIIDR	None

5.3.6 FMU_SMEN, Safety Mechanism Enable Register

Enables or disables particular protection mechanisms for a specified MHU block.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF00

Bit descriptions

Figure 5-6: MHU_FMU_REGISTER_BLOCK_FMU_SMEN bit assignments

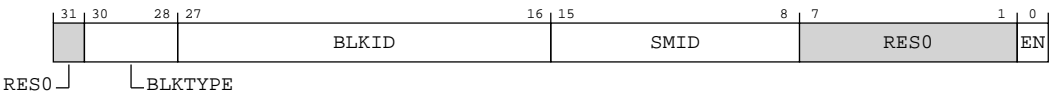


Table 5-14: FMU_SMEN bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30:28]	BLKTYPE	MHU block type identifier. 0b000 Sender block. 0b001 Receiver block. 0b010 FMU block.	–
[27:16]	BLKID	MHU block identifier. Selects the specific block type instance. 0b00000000000000 Identifier 0.	–
[15:8]	SMID	Protection Mechanism identifier.	–
[7:1]	RES0	Reserved	RES0
[0]	EN	Protection Mechanism enable or disable. 0b0 Disable protection mechanism. 0b1 Enable protection mechanism.	–

Accessibility

Table 5-15: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF00	FMU_SMEN	None

5.3.7 FMU_SMERR, Safety Mechanism Error Register

Inserts an error into the specified Safety Mechanism inside an MHU block.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF04

Bit descriptions

Figure 5-7: MHU_FMU_REGISTER_BLOCK_FMU_SMERR bit assignments

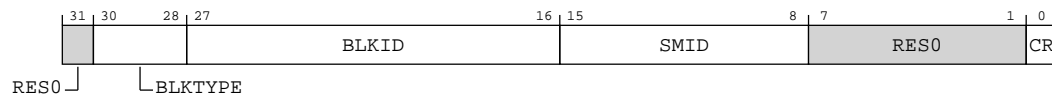


Table 5-16: FMU_SMERR bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30:28]	BLKTYPE	MHU block type identifier. 0b000 Sender block. 0b001 Receiver block. 0b010 FMU block.	–
[27:16]	BLKID	MHU block identifier. Selects the specific block type instance. 0b00000000000000 Identifier 0.	–
[15:8]	SMID	Protection Mechanism identifier.	–
[7:0]	RES0	Reserved	RES0

Accessibility

Table 5-17: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF04	FMU_SMERR	None

5.3.8 FMU_SMCR, Safety Mechanism Set Criticality Register

Sets the Protection Mechanism criticality.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF08

Bit descriptions

Figure 5-8: MHU_FMU_REGISTER_BLOCK_FMU_SMCR bit assignments

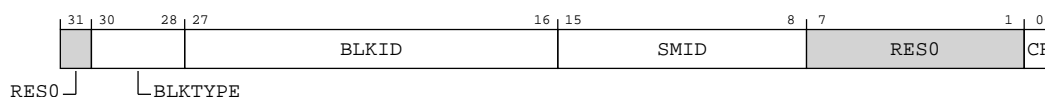


Table 5-18: FMU_SMCR bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30:28]	BLKTYPE	MHU block type identifier. 0b000 Sender block. 0b001 Receiver block. 0b010 FMU block.	–
[27:16]	BLKID	MHU block identifier. Selects the specific block type instance. 0b00000000000000 Identifier 0.	–
[15:8]	SMID	Protection Mechanism identifier.	–
[7:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	CR	Set Protection Mechanism criticality. 0b0 Set the protection mechanism as non-critical. 0b1 Set the protection mechanism as critical.	–

Accessibility

Table 5-19: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF08	FMU_SMCR	None

5.3.9 FMU_SMWR, Safety Mechanism Page Write Register

Performs a page write and page read back access for the PAGEID. The write data used is taken from FMU_SMWDATA and the read back of the written data goes into FMU_SMRDATA.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF0C

Bit descriptions

Figure 5-9: MHU_FMU_REGISTER_BLOCK_FMU_SMWR bit assignments

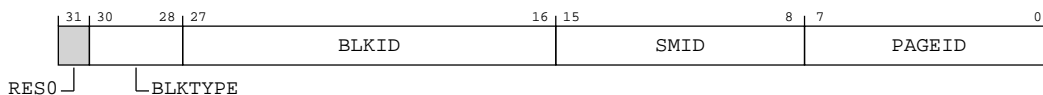


Table 5-20: FMU_SMWR bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[30:28]	BLKTYPE	MHU block type identifier. 0b000 Sender block. 0b001 Receiver block. 0b010 FMU block.	–
[27:16]	BLKID	MHU block identifier. Selects the specific block type instance. 0b00000000000000 Identifier 0.	–
[15:8]	SMID	Protection Mechanism identifier.	–
[7:0]	PAGEID	Page identifier for data to be written to. 0b00000000 Page 0 - Protection Mechanism enable, criticality and insertion. 0b00000001 Page 1 - Protection Mechanism timeout values. 0b00000010 Page 2 - Protection Mechanism CRC error types.	–

Accessibility

Table 5-21: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF0C	FMU_SMWR	None

5.3.10 FMU_SMWDATA, Safety Mechanism Page Write Data Register

Provides the Protection Mechanism page write data when FMU_SMWR is written.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF10

Bit descriptions

Figure 5-10: MHU_FMU_REGISTER_BLOCK_FMU_SMWDATA bit assignments



Table 5-22: FMU_SMWDATA bit descriptions

Bits	Name	Description	Reset
[31:0]	DATA	<p>Protection Mechanism write data of the page indicated by FMU_SMWR.PAGEID.</p> <p>The write data contents depend on the page being accessed.</p> <p>For Page 0 accesses (available for all protection mechanisms):</p> <ul style="list-style-type: none"> bit [4]: Inserted. Inserts or clears an error for testing purposes. bit [2]: Critical. Sets the criticality of a protection mechanism. bit [1]: Enable. Enables or disables a protection mechanism. <p>For Page 1 accesses (available for AXI5-Stream or ACE5-Lite protection):</p> <ul style="list-style-type: none"> bits [31:16]: SendTime. Sets the duration when it becomes high priority for the block to send a ping or ping acknowledgement packet: 4 x (SendTime + 1) - 1 cycles. bits [15:0]: ErrTime. Sets the duration when the block detects a timeout error, because a ping or ping acknowledgement packet is missing: 64 x (ErrTime + 1) - 1 cycles. Arm recommends that the time to error is at least 4 times longer than the time to send a ping or a ping acknowledgement packet. However, interconnect delays or the different frequencies of both domains might require this recommendation to be longer. <p>For Page 2 accesses (available for AXI5-Stream or ACE5-Lite protection):</p> <ul style="list-style-type: none"> bit [1]: crc_timeout_err. Set to 1, to clear a logged CRC timeout error. bit [1]: crc_checksum_err. Set to 1, to clear a logged CRC checksum error. 	32 {x}

Accessibility

Table 5-23: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF10	FMU_SMWDATA	None

5.3.11 FMU_SMRD, Safety Mechanism Page Read Register

Performs a page read access for the PAGEID. The read data is returned in FMU_SMRDATA.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF14

Bit descriptions

Figure 5-11: MHU_FMU_REGISTER_BLOCK_FMU_SMRD bit assignments

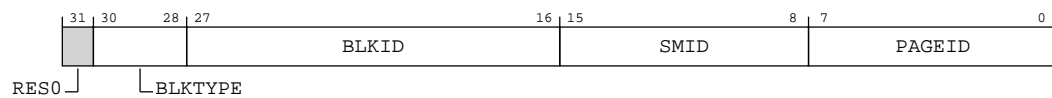


Table 5-24: FMU_SMRD bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30:28]	BLKTYPE	MHU block type identifier. 0b000 Sender block. 0b001 Receiver block. 0b010 FMU block.	–
[27:16]	BLKID	MHU block identifier. Selects the specific block type instance. 0b00000000000000 Identifier 0.	–
[15:8]	SMID	Protection Mechanism identifier.	–
[7:0]	PAGEID	Page identifier for data being read from. 0b00000000 Page 0 - Protection Mechanism enable, criticality and insertion. 0b00000001 Page 1 - Protection Mechanism timeout values. 0b00000010 Page 2 - Protection Mechanism CRC error types.	–

Accessibility

Table 5-25: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF14	FMU_SMRD	None

5.3.12 FMU_SMRDATA, Safety Mechanism Page Read Data Register

Returns the Protection Mechanism page read data when FMU_SMRD or FMU_SMWR is written.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF18

Bit descriptions

Figure 5-12: MHU_FMU_REGISTER_BLOCK_FMU_SMRDATA bit assignments



Table 5-26: FMU_SMRDATA bit descriptions

Bits	Name	Description	Reset
[31:0]	DATA	<p>Protection Mechanism read data of the page indicated by FMU_SMRD.PAGEID or FMU_SMWR.PAGEID.</p> <p>The read data contents depend on the page being accessed.</p> <p>For Page 0 accesses (available for all protection mechanisms):</p> <ul style="list-style-type: none"> bit [4]: Inserted. Returns error insertion status. bit [2]: Critical. Returns the protection mechanism criticality. bit [1]: Enable. Returns whether protection mechanism is enabled (when SMID != 255) or whether the error wire outputs are enabled (SMID == 255). <p>For Page 1 accesses (available for AXI5-Stream or ACE5-Lite protection):</p> <ul style="list-style-type: none"> bits [31:16]: SendTime. Returns the duration when it becomes high priority for the block to send a ping or ping acknowledgement packet: <ul style="list-style-type: none"> 4 x (SendTime + 1) - 1 cycles. bits [15:0]: ErrTime. Returns the duration when the block detects a timeout error, because a ping or ping acknowledgement packet is missing: <ul style="list-style-type: none"> 64 x (ErrTime + 1) - 1 cycles. Arm recommends that the time to error is at least 4 times longer than the time to send a ping or a ping acknowledgement packet. However, interconnect delays or the different frequencies of both domains might require this recommendation to be longer. <p>For Page 2 accesses (available for AXI5-Stream or ACE5-Lite protection):</p> <ul style="list-style-type: none"> bit [1]: crc_timeout_err. When set to 1, it indicates that a CRC timeout error has occurred. bit [1]: crc_checksum_err. When set to 1, it indicates that a CRC checksum error has occurred. 	0x00000000

Accessibility

Table 5-27: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF18	FMU_SMRDATA	None

5.3.13 FMU_STATUS, FMU Status Register

Monitors whether there are outstanding FMU accesses waiting for responses.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF1C

Bit descriptions

Figure 5-13: MHU_FMU_REGISTER_BLOCK_FMU_STATUS bit assignments

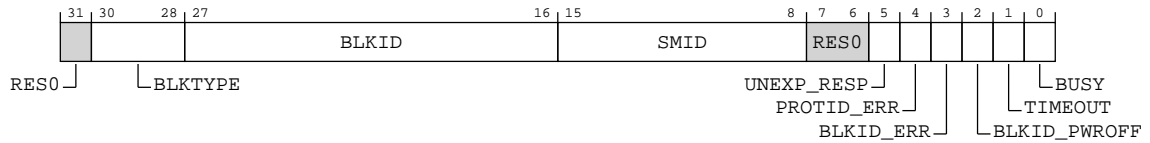


Table 5-28: FMU_STATUS bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30:28]	BLKTYPE	MHU block type identifier from the last response. 0b000 Sender block. 0b001 Receiver block. 0b010 FMU block.	0b000
[27:16]	BLKID	MHU block identifier from the last response. Selects the specific block type instance. 0b00000000000000 Identifier 0.	0x000
[15:8]	SMID	Protection Mechanism identifier from the last response.	0x00
[7:6]	RES0	Reserved	RES0
[5]	UNEXP_RESP	The last received response was unexpected. This may have been caused by a timeout.	0b0
[4]	PROTID_ERR	Set if a write or read to a block attempted to access an invalid PROTID or PAGEID. For reads this indicates FMU_SMRDATA is invalid.	0b0
[3]	BLKID_ERR	BLKID does not exist.	0b0
[2]	BLKID_PWROFF	Last command failed due to BLKID powered-off.	0b0
[1]	TIMEOUT	The last FMU response timed out. See FMU_TIMEOUT for information on the timeout duration.	0b0
[0]	BUSY	Indicates if the FMU is busy. 0b0 FMU is not busy. 0b1 FMU is busy.	0b0

Accessibility

Table 5-29: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0x0F1C	FMU_STATUS	None

5.3.14 FMU_KEY, FMU Key Register

Used to receiver the unlock key that is required for writes to FMU registers to be successful. This mechanism does not affect ability to perform FMU reads.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF20

Bit descriptions

Figure 5-14: MHU_FMU_REGISTER_BLOCK_FMU_KEY bit assignments



Table 5-30: FMU_KEY bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	KEY	<p>Writing the correct key to this field enables the next write to any other writable FMU register to succeed.</p> <p>The register file is unlocked when a write to FMU_KEY occurs that satisfies all of the following conditions:</p> <ul style="list-style-type: none">Is SecureIs for 32 bitsThe bottom 8 bits are 0xBE <p>If the register file is unlocked, the FMU_KEY register reads as 0x00000BE. Otherwise, the FMU_KEY register reads as 0x00000000</p> <p>The FMU_KEY register automatically locks after most Secure write access. The FMU_KEY register locks even if the Secure write is ignored, for example, if it is a write to invalid address. However, the FMU_KEY does not lock automatically for Secure writes to the upper 32-bits of the 64-bit RAS registers FMU_ERR<n>CTLR and FMU_ERR<n>STATUS.</p>	0x00

Accessibility

Table 5-31: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF20	FMU_KEY	None

5.3.15 FMU_TIMEOUT, FMU Timeout Duration Register

Defines the duration of the timeout period before TIMEOUT is reported in FMU_STATUS

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF24

Bit descriptions

Figure 5-15: MHU_FMU_REGISTER_BLOCK_FMU_TIMEOUT bit assignments



Table 5-32: FMU_TIMEOUT bit descriptions

Bits	Name	Description	Reset
[31:0]	DURATION	Timeout count duration in clock cycles. The initial value on reset is all 1s to indicate the longest possible timeout allowed.	0xFFFFFFFF

Accessibility

Table 5-33: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF24	FMU_TIMEOUT	None

5.3.16 FMU_ERRUPDATE, FMU Error Update Register

Forces the record pair FMU_ERR<n>STATUS (indices n and n+1) to be updated with all error state reported through this record pair.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF28

Bit descriptions

Figure 5-16: MHU_FMU_REGISTER_BLOCK_FMU_ERRUPDATE bit assignments

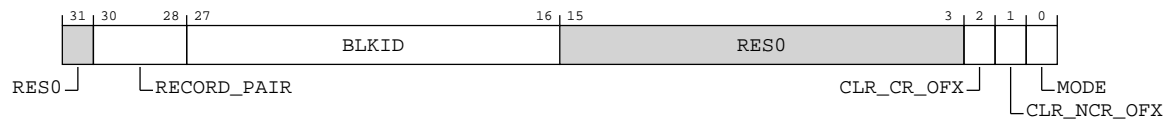


Table 5-34: FMU_ERRUPDATE bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30:28]	RECORD_PAIR	Identifier of error record pair to update. Updated records will be records n and n+1 where $n=2*RECORD_PAIR$. 0b000 Sender block records. 0b001 Receiver block records. 0b010 FMU block records.	–
[27:16]	BLKID	MHU starting block identifier for error update. 0b00000000000000 Identifier 0.	–
[15:3]	RES0	Reserved	RES0
[2]	CLR_CR_OFX	Clears the critical record's FMU_ERR<n>STATUS.OFX bit. ($n = RECORD_PAIR*2$).	–
[1]	CLR_NCR_OFX	Clears the non-critical record's FMU_ERR<n>STATUS.OFX bit. ($n = RECORD_PAIR*2+1$).	–

Bits	Name	Description	Reset
[0]	MODE	BLKID update mode 0b0 Increment BLKID until BLKID_INVALID returned or BLKID wraps. 0b1 Increment BLKID until powered down block found or BLKID wraps.	–

Accessibility

Table 5-35: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xF28	FMU_ERRUPDATE	None

5.3.17 FMU_FCTLR, FMU Feature Control Register

Controls additional clock gating functionality.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xF2C

Bit descriptions

Figure 5-17: MHU_FMU_REGISTER_BLOCK_FMU_FCTLR bit assignments

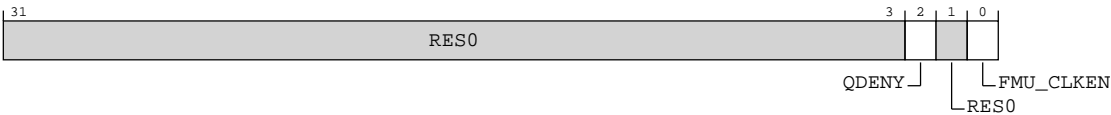


Table 5-36: FMU_FCTLR bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0

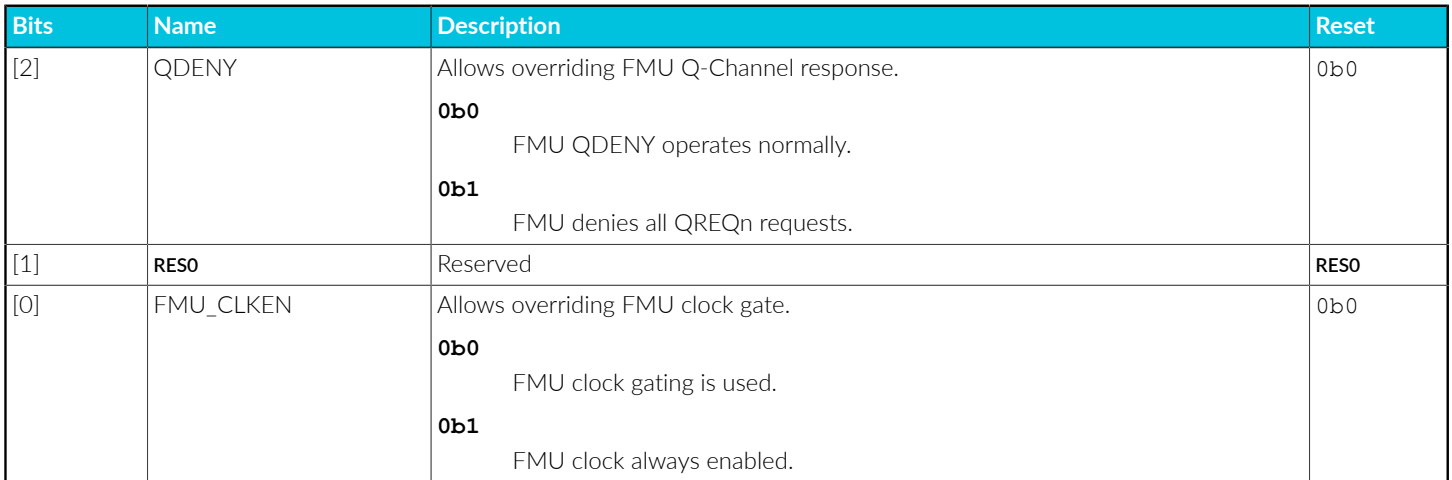


Table 5-38: FMU_ERRDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-39: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFBC	FMU_ERRDEVARCH	None

5.3.19 FMU_ERRDEVID, Device Configuration Register

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFC8

Bit descriptions

Figure 5-19: MHU_FMU_REGISTER_BLOCK_FMU_ERRDEVID bit assignments

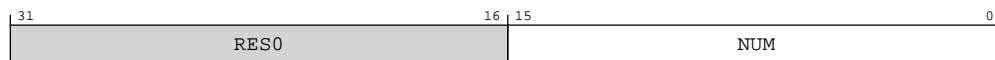


Table 5-40: FMU_ERRDEVID bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:0]	NUM	Highest numbered index of the error records in this group, plus one.	0x6

Accessibility

Table 5-41: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFC8	FMU_ERRDEVID	None

5.3.20 FMU_PIDR4, MHU FMU Peripheral ID 4 Register

Returns byte[4] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFD0

Bit descriptions

Figure 5-20: MHU_FMU_REGISTER_BLOCK_FMU_PIDR4 bit assignments

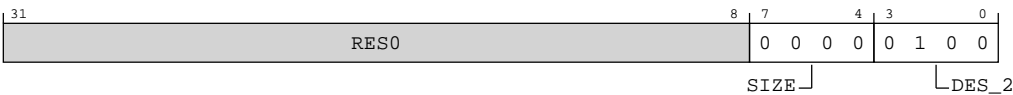


Table 5-42: FMU_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	<div>Size of the component</div> <div>0b0000</div> <div>The size of the component must be identified using a combination of the following registers:</div> <ul style="list-style-type: none">Peripheral ID 0-7 registersComponent ID 0-3 registers<x>_FEAT_SPT0 and <x>_FEAT_SPT1 registers	0b0000
[3:0]	DES_2	<div>JEP106 continuation code</div> <div>0b0100</div>	0b0100

Accessibility

Table 5-43: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFD0	FMU_PIDR4	None

5.3.21 FMU_PIDR5, MHU FMU Peripheral ID 5 Register

Returns byte[5] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFD4

Bit descriptions

Figure 5-21: MHU_FMU_REGISTER_BLOCK_FMU_PIDR5 bit assignments

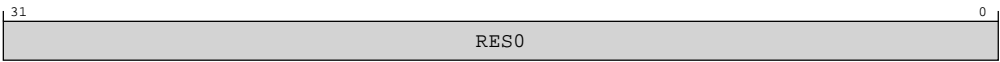


Table 5-44: FMU_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-45: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFD4	FMU_PIDR5	None

5.3.22 FMU_PIDR6, MHU FMU Peripheral ID 6 Register

Returns byte[6] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFD8

Bit descriptions

Figure 5-22: MHU_FMU_REGISTER_BLOCK_FMU_PIDR6 bit assignments

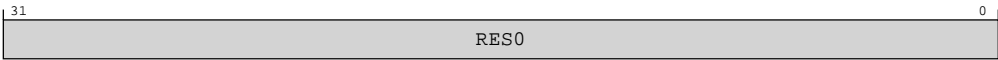


Table 5-46: FMU_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-47: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFD8	FMU_PIDR6	None

5.3.23 FMU_PIDR7, MHU FMU Peripheral ID 7 Register

Returns byte[7] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFDC

Bit descriptions

Figure 5-23: MHU_FMU_REGISTER_BLOCK_FMU_PIDR7 bit assignments

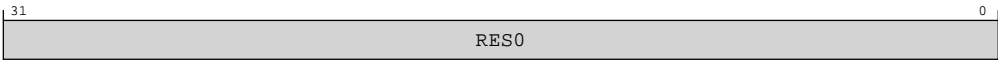


Table 5-48: FMU_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-49: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFDC	FMU_PIDR7	None

5.3.24 FMU_PIDR0, MHU FMU Peripheral ID 0 Register

Returns byte[0] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFE0

Bit descriptions

Figure 5-24: MHU_FMU_REGISTER_BLOCK_FMU_PIDR0 bit assignments



Table 5-50: FMU_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Bits 7:0 of the Part ID for the implementation of the MHU	The reset value depends on whether the FMU is located in the MHU Sender or MHU Receiver. <ul style="list-style-type: none"> 0x9C - MHU Sender FMU 0x9D - MHU Receiver FMU

Accessibility

Table 5-51: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFE0	FMU_PIDR0	None

5.3.25 FMU_PIDR1, MHU FMU Peripheral ID 1 Register

Returns byte[1] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFE4

Bit descriptions

Figure 5-25: MHU_FMU_REGISTER_BLOCK_FMU_PIDR1 bit assignments



Table 5-52: FMU_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	DES_0	Bits 3:0 of the JEP106 identification code 0b1011	0b1011
[3:0]	PART_1	Bits 11:8 of the Part ID for the implementation of the MHU	0x4

Accessibility

Table 5-53: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFE4	FMU_PIDR1	None

5.3.26 FMU_PIDR2, MHU FMU Peripheral ID 2 Register

Returns byte[2] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFE8

Bit descriptions

Figure 5-26: MHU_FMU_REGISTER_BLOCK_FMU_PIDR2 bit assignments

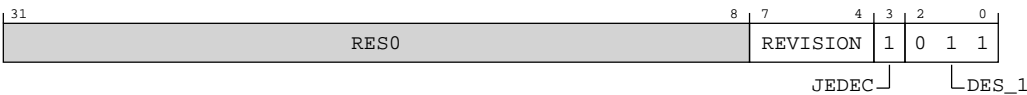


Table 5-54: FMU_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	The REVISION field is an incremental value starting at 0x0 for the first design of a component. The value is increased by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision.	0x0
[3]	JEDEC	Must be 0b1 to indicate that a JEDEC-assigned value is used. 0b1	0b1
[2:0]	DES_1	JEP106 identification bits [6:4] 0b011	0b011

Accessibility

Table 5-55: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFE8	FMU_PIDR2	None

5.3.27 FMU_PIDR3, MHU FMU Peripheral ID 3 Register

Returns byte[3] of the peripheral ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFEC

Bit descriptions

Figure 5-27: MHU_FMU_REGISTER_BLOCK_FMU_PIDR3 bit assignments



Table 5-56: FMU_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	<p>The REVAND field indicates minor errata fixes specific to this design, for example metal fixes after implementation. Usually this field is zero.</p> <p>Together with PIDR2.REVISION, PIDR3.REVAND forms the revision number of the component.</p>	0x0
[3:0]	CMOD	<p>Customer Modified.</p> <p>If the component is reusable IP, the CMOD field indicates whether the customer has modified the behavior of the component.</p> <p>Arm recommends that the user or debugger reads the documentation for the component to determine the modifications that are made to the component.</p> <p>For any two components with the same values of the Peripheral ID 0-7 and Component ID 0-3 registers</p> <ul style="list-style-type: none"> If the value of the CMOD fields of both components equals zero, the components are identical If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have been subjected to the same modifications. If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same values of the Peripheral ID 0-7 and Component ID 0-3 registers 	0x0

Accessibility

Table 5-57: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFEC	FMU_PIDR3	None

5.3.28 FMU_CIDR0, MHU FMU Component ID 0 Register

Returns byte[0] of the component ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFF0

Bit descriptions

Figure 5-28: MHU_FMU_REGISTER_BLOCK_FMU_CIDR0 bit assignments

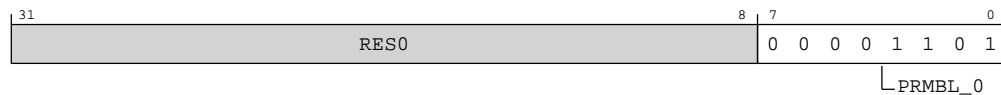


Table 5-58: FMU_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble, segment 0 0b00001101	0x0D

Accessibility

Table 5-59: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFF0	FMU_CIDR0	None

5.3.29 FMU_CIDR1, MHU FMU Component ID 1 Register

Returns byte[1] of the component ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFF4

Bit descriptions

Figure 5-29: MHU_FMU_REGISTER_BLOCK_FMU_CIDR1 bit assignments

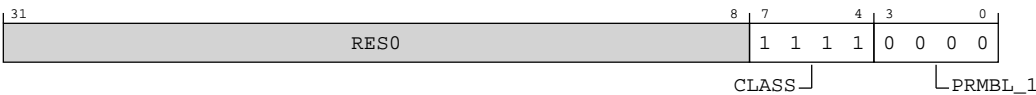


Table 5-60: FMU_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component Class 0b1111 CoreLink, PrimeCell, or system component with no standardized register layout, for backwards compatibility.	0b1111
[3:0]	PRMBL_1	Preamble, segment 1 0b0000	0b0000

Accessibility

Table 5-61: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFF4	FMU_CIDR1	None

5.3.30 FMU_CIDR2, MHU FMU Component ID 2 Register

Returns byte[2] of the component ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFF8

Bit descriptions

Figure 5-30: MHU_FMU_REGISTER_BLOCK_FMU_CIDR2 bit assignments

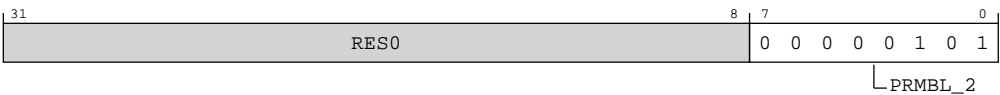


Table 5-62: FMU_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble, segment 2 0b00000101	0x05

Accessibility

Table 5-63: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFF8	FMU_CIDR2	None

5.3.31 FMU_CIDR3, MHU FMU Component ID 3 Register

Returns byte[3] of the component ID for MHU FMU page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHU FMU Register Block

Register offset

0xFFC

Bit descriptions

Figure 5-31: MHU_FMU_REGISTER_BLOCK_FMU_CIDR3 bit assignments

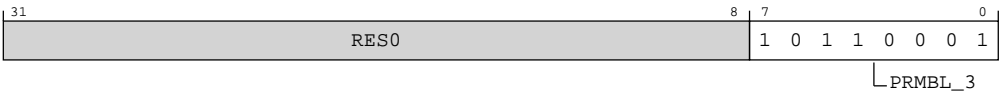


Table 5-64: FMU_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble, segment 3 0b10110001	0xB1

Accessibility

Table 5-65: Accessibility

Component	Offset	Instance	Range
MHU FMU Register Block	0xFFC	FMU_CIDR3	None

5.4 MHU Sender registers

The MHU Sender registers include the register blocks that are part of the MHU Sender.

The offsets of the blocks, within the MHU Sender are IMPLEMENTATION DEFINED. However, we recommend the following offsets.

- When TrustZone (TZE) is implemented for the MHUS
 - 0x0_0000 - Sender Security Control (SSC) registers
 - 0x1_0000 - Postbox (PBX) registers
 - 0x2_0000 - MHU Sender RAS (SRAS) registers
- When TZE is not implemented for the MHUS
 - 0x0_0000 - PBX registers
 - 0x1_0000 - SRAS registers

5.4.1 MHU Sender Security Control register summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MHU Sender Security Control (MHUS.SSC) registers .

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, see the individual field resets documented on the register description pages or the [Message Handling Unit Architecture version 3.0](#).

Table 5-66: MHUS.SSC register summary

Offset	Name	Type	Reset	Width	Description
0x0000	SSC_BLK_ID	RO	See individual bit resets.	32-bit	Sender Security Control Block Identifier Register
0x0010	SSC_FEAT_SPT0	RO	See individual bit resets.	32-bit	Sender Security Feature Support 0 Register
0x0014	SSC_FEAT_SPT1	RO	See individual bit resets.	32-bit	Sender Security Feature Support 1 Register
0x0020	SSC_DBCH_CFG0	RO	See individual bit resets.	32-bit	Sender Security Control Doorbell Channel Configuration 0 Register
0x0030	SSC_FFCH_CFG0	RO	See individual bit resets.	32-bit	Sender Security Control FIFO Channel Configuration 0 Register
0x0040	SSC_FCH_CFG0	RO	See individual bit resets.	32-bit	Sender Security Control Fast Channel Configuration 0 Register
(4 * n) + 0x0110	SSC_PBX_SG<n>	RW	See individual bit resets.	32-bit	Sender Postbox Security Group n Register
0x0FC8	SSC_IIDR	RO	See individual bit resets.	32-bit	Sender Security Implementer Identification Register
0x0FCC	SSC_AIDR	RO	See individual bit resets.	32-bit	Sender Security Architecture Identification Register
0x0FD0	SSC_PIDR4	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 4 Register
0x0FD4	SSC_PIDR5	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 5 Register
0x0FD8	SSC_PIDR6	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 6 Register
0x0FDC	SSC_PIDR7	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 7 Register
0x0FE0	SSC_PIDR0	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 0 Register
0x0FE4	SSC_PIDR1	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 1 Register
0x0FE8	SSC_PIDR2	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 2 Register
0x0FEC	SSC_PIDR3	RO	See individual bit resets.	32-bit	Sender Security Peripheral ID 3 Register

Offset	Name	Type	Reset	Width	Description
0x0FF0	SSC_CIDR0	RO	See individual bit resets.	32-bit	Sender Security Component ID 0 Register
0x0FF4	SSC_CIDR1	RO	See individual bit resets.	32-bit	Sender Security Component ID 1 Register
0x0FF8	SSC_CIDR2	RO	See individual bit resets.	32-bit	Sender Security Component ID 2 Register
0x0FFC	SSC_CIDR3	RO	See individual bit resets.	32-bit	Sender Security Component ID 3 Register
0xF000	SSC_ACTRL	RW	See individual bit resets.	32-bit	MHUS Access Control Register

5.4.1.1 SSC_BLK_ID, Sender Security Control Block Identifier Register

Identifies the block as a Sender Security Control.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_BLK_ID are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0000

Bit descriptions

Figure 5-32: MHUS.SSC_SSC_BLK_ID bit assignments

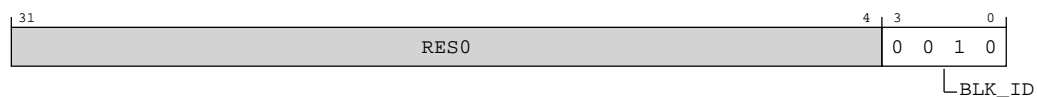


Table 5-67: SSC_BLK_ID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	BLK_ID	Block Identifier Identifies the block as a Sender Security Control 0b0010 Sender Security Control	0b0010

Accessibility

Table 5-68: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0000	SSC_BLK_ID	None

5.4.1.2 SSC_FEAT_SPT0, Sender Security Feature Support 0 Register

Returns information on supported MHU features

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_FEAT_SPT0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0010

Bit descriptions

Figure 5-33: MHUS.SSC_SSC_FEAT_SPT0 bit assignments

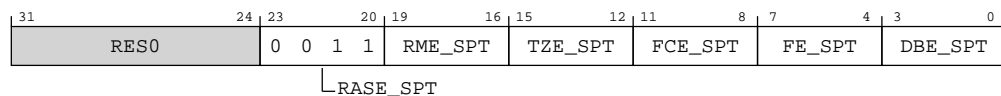


Table 5-69: SSC_FEAT_SPT0 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:20]	RASE_SPT	Reliability, Availability and Serviceability Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0011 MHU implements the RAS extension and follows the recommendations in the "Recommend implementation of RAS using Arm RAS extensions" section of Message Handling Unit Architecture version 3.0 .	0b0011

Bits	Name	Description	Reset
[19:16]	RME_SPT	<p>Realm Management Extension Support</p> <p>The value of this field depends on the implementation of the MHU and an optional reset time sampled input LEGACY_TZ_EN. The field can take one of the following values:</p> <p>0b0000 MHU does not implement the Realm Management extension</p> <p>0b0001 MHU implements Realm Management extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p> <p>When RME is implemented, for the MHU component, there can be a LEGACY_TZ_EN tie-off signal present on this MHU component.</p> <p>The value of the LEGACY_TZ_EN tie-off signal is sampled at reset of the MHU component which the tie-off signal is associated with.</p> <p>When the sampled value of the tie-off signal is 0b1 the value of this field is always 0x0, otherwise the value of this field is dependent on whether RME is implemented for this MHU component.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>
[15:12]	TZE_SPT	<p>TrustZone Extension Support</p> <p>The value of this field depends on the implementation of the MHU and can take one of the following values:</p> <p>0b0000 MHU does not implement the TrustZone extension</p> <p>0b0001 MHU implements TrustZone extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>

Bits	Name	Description	Reset
[11:8]	FCE_SPT	Fast Channel Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Fast Channel extension 0b0001 MHU implements Fast Channel extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[7:4]	FE_SPT	FIFO Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the FIFO extension 0b0001 MHU implements FIFO extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[3:0]	DBE_SPT	Doorbell Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Doorbell extension 0b0001 MHU implements Doorbell extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.

Accessibility

Table 5-70: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0010	SSC_FEAT_SPT0	None

5.4.1.3 SSC_FEAT_SPT1, Sender Security Feature Support 1 Register

Returns information on supported MHU features

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_FEAT_SPT1 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset
0x0014

Bit descriptions

Figure 5-34: MHUS.SSC_SSC_FEAT_SPT1 bit assignments

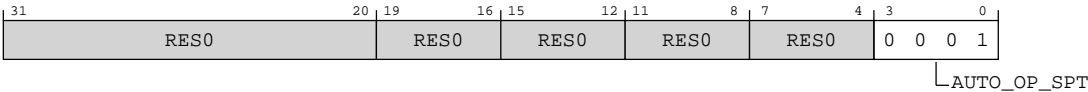


Table 5-71: SSC_FEAT_SPT1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	AUTO_OP_SPT	Auto Op protocol support. For more information about the Auto Op protocol, see the Message Handling Unit Architecture version 3.0 . The value of this field is set for MHU-320AE: 0b0001 Auto Op(Full) is implemented	0b0001

Accessibility

Table 5-72: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0014	SSC_FEAT_SPT1	None

5.4.1.4 SSC_DBCH_CFG0, Sender Security Control Doorbell Channel Configuration 0 Register

Returns doorbell channel configuration information

Configurations

This register is present only when TZE is implemented for the MHUS and DBE is implemented. Otherwise, direct accesses to SSC_DBCH_CFG0 are RAZ/WI.

Attributes

Width
32

Component
MHUS.SSC

Register offset
0x0020

Bit descriptions

Figure 5-35: MHUS.SSC_SSC_DBCH_CFG0 bit assignments

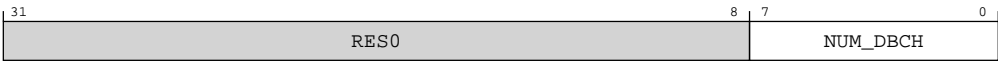


Table 5-73: SSC_DBCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	NUM_DBCH	Number of Doorbell Channels 0b00000000..0b01111111 Number of DBCH is N+1, where N is the value of this field	8 {x}

Accessibility

Table 5-74: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0020	SSC_DBCH_CFG0	None

5.4.1.5 SSC_FFCH_CFG0, Sender Security Control FIFO Channel Configuration 0 Register

Returns FIFO channel configuration information

Configurations

This register is present only when TZE is implemented for the MHUS and FE is implemented. Otherwise, direct accesses to SSC_FFCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0030

Bit descriptions

Figure 5-36: MHUS.SSC_FFCH_CFG0 bit assignments

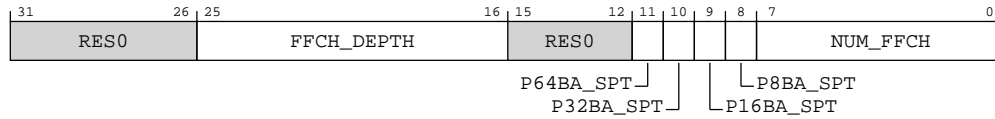


Table 5-75: SSC_FFCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25:16]	FFCH_DEPTH	FIFO Channel Depth 0b0000000000..0b1111111111 FIFO depth is N+1 bytes, where N is the value of this field	The reset value for this field depends on the MHU configuration.
[15:12]	RES0	Reserved	RES0
[11]	P64BA_SPT	Postbox 64bit Access Support Whether the implementation of the MHU supports 64bit accesses to the PFFCW<n>_PAY register 0b0 64bit accesses are not supported 0b1 64bit accesses are supported Accesses must be aligned to an 64bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.
[10]	P32BA_SPT	Postbox 32bit Access Support Whether the implementation of the MHU supports 32bit accesses to the PFFCW<n>_PAY register 0b0 32bit accesses are not supported 0b1 32bit accesses are supported Accesses must be aligned to an 32bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[9]	P16BA_SPT	<p>Postbox 16bit Access Support</p> <p>Whether the implementation of the MHU supports 16bit accesses to the PFFCW<n>_PAY register</p> <p>0b0 16bit accesses are not supported</p> <p>0b1 16bit accesses are supported</p> <p>Accesses must be aligned to an 16bit boundary</p>	0x0
[8]	P8BA_SPT	<p>Postbox 8bit Access Support</p> <p>Whether the implementation of the MHU supports 8bit accesses to the PFFCW<n>_PAY register</p> <p>0b0 8bit accesses are not supported</p> <p>0b1 8bit accesses are supported</p> <p>Accesses must be aligned to an 8bit boundary</p>	0x0
[7:0]	NUM_FFCH	<p>Number of FIFO Channels</p> <p>The number of FIFO Channels in the Postbox.</p> <p>0b00000000..0b00111111 Number of FIFO is N+1, where N is the value of this field</p>	The reset value for this field depends on the MHU configuration.

Accessibility

Table 5-76: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0030	SSC_FFCH_CFG0	None

5.4.1.6 SSC_FCH_CFG0, Sender Security Control Fast Channel Configuration 0 Register

Returns fast channel configuration information

Configurations

This register is present only when TZE is implemented for the MHUS and FCE is implemented. Otherwise, direct accesses to SSC_FCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0040

Bit descriptions

Figure 5-37: MHUS.SSC_SSC_FCH_CFG0 bit assignments

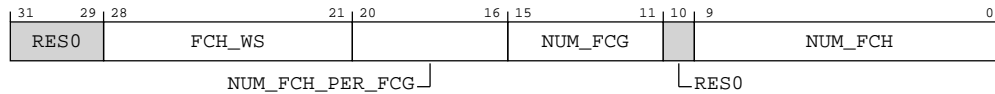


Table 5-77: SSC_FCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:29]	RES0	Reserved	RES0
[28:21]	FCH_WS	Fast Channel Word Size Number of bits each Fast Channel implements. The value must be the same as MBX_FCH_CFG0.FCH_WS. 0b00100000 Fast Channel word size is 32-bits 0b01000000 Fast Channel word size is 64-bits	The reset values can be the following: 0b00100000, 0b01000000, respective to the value.
[20:16]	NUM_FCH_PER_FCG	Number of Fast Channels per Fast Channel Group for the Postbox 0b00000..0b11111 Number of Fast Channels per Fast Channel Group is N+1, where N is the value of this field The number of Fast Channels in the last Fast Channel Group can be less than the value in this field if the value of NUM_FCH is not a multiple of the number of Fast Channels per Fast Channel Group. All other values are Reserved	The reset value for this field depends on the MHU configuration.
[15:11]	NUM_FCG	Number of Fast Channel Groups for the MHUS. 0b00000..0b11111 The number of Fast Channel Groups is N+1, where N is the value of this field The legal values for this field are 0-31	The reset value for this field depends on the MHU configuration.
[10]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[9:0]	NUM_FCH	<p>Number of Fast Channels in the MHUS.</p> <p>FCH_WS == 0x40</p> <p>0b0000000000..0b0111111111</p> <p>The number of FCH is N+1, where N is the value of this field</p> <p>FCH_WS == 0x20</p> <p>0b0000000000..0b1111111111</p> <p>The number of FCH is N+1, where N is the value of this field</p>	The reset value for this field depends on the MHU configuration.

Accessibility

Table 5-78: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0040	SSC_FCH_CFG0	None

5.4.1.7 SSC_PBX_SG<n>, Sender Postbox Security Group n Register, n = 0

Returns security configuration information. The bit descriptions for this register depend on whether the RME or TZE are implemented for the MHU Sender.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PBX_SG<n> are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

(4 * n) + 0x0110

Bit descriptions

When RME is implemented for the MHUS and sampled value of MHUS LEGACY_TZ_EN is 0b0, the bit assignments are as follows:

Figure 5-38: MHUS.SSC_SSC_PBX_SG<n> bit assignments

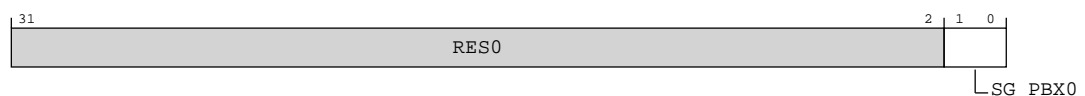


Table 5-79: SSC_PBX_SG<n> bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	SG_PBX<m>, bit[m], where m = 0	Security Group for Postbox 0b00 Postbox is assigned to the Secure Security Group 0b01 Postbox is assigned to the Non-secure Security Group 0b10 Postbox is assigned to the Root Security Group 0b11 Postbox is assigned to the Realm Security Group	0b10

When RME is not implemented for the MHUS or RME is implemented for the MHUS and sampled value of MHUS LEGACY_TZ_EN is 0b1, the bit assignments are as follows:

Figure 5-39: MHUS.SSC_SSC_PBX_SG<n> bit assignments

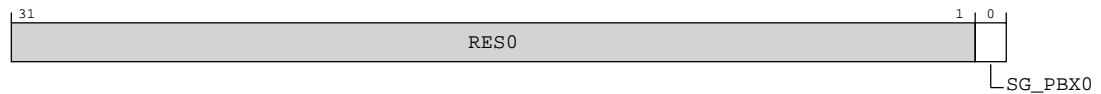


Table 5-80: SSC_PBX_SG<n> bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	SG_PBX<m>, bit[m], where m = 0	Security Group for Postbox 0b0 Postbox is assigned to the Secure Security Group 0b1 Postbox is assigned to the Non-secure Security Group	0b0

Accessibility

Table 5-81: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	(4 * n) + 0x0110	SSC_PBX_SG<n>	None

5.4.1.8 SSC_IIDR, Sender Security Implementer Identification Register

This field provides information on the Implementer of the MHU

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_IIDR are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FC8

Bit descriptions

Figure 5-40: MHUS.SSC_SSC_IIDR bit assignments

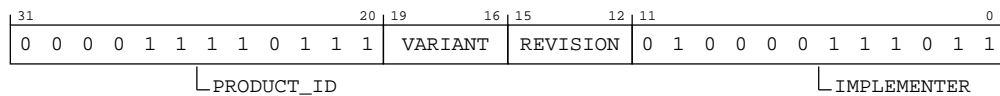


Table 5-82: SSC_IIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	PRODUCT_ID	Product ID of the MHU implementation 0b000011110111	0x0F7
[19:16]	VARIANT	Variant or Major revision of the MHU implementation	The reset value depends on the product version used. • 0x0 - r0
[15:12]	REVISION	Revision or minor version of the MHU implementation	The reset value depends on the product version used. • 0x0 - p0 • 0x1 - p1
[11:0]	IMPLEMENTER	Implementer ID Contains the JEP106 identification information as follows: <ul style="list-style-type: none"> 11:8 - JEP106 continuation code of implementer 7 - Always 0 6:0 - JEP106 identity code of implementer 0b010000111011	0x43B

Accessibility

Table 5-83: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FC8	SSC_IIDR	None

5.4.1.9 SSC_AIDR, Sender Security Architecture Identification Register

Provides information on the implemented MHU architecture

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_AIDR are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FCC

Bit descriptions

Figure 5-41: MHUS.SSC_SSC_AIDR bit assignments

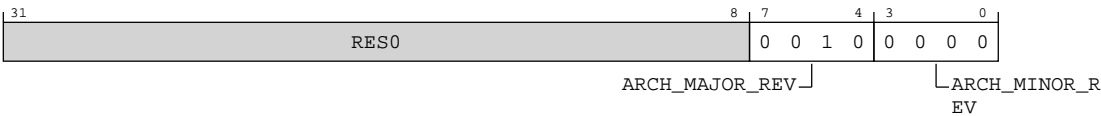


Table 5-84: SSC_AIDR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	ARCH_MAJOR_REV	MHU Architecture Major Revision 0b0010 MHUv3 All other values are Reserved	0b0010
[3:0]	ARCH_MINOR_REV	MHU Architecture Minor Revision 0b0000 Minor revision 0 of the major architecture All other values are Reserved	0b0000

Accessibility

Table 5-85: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FCC	SSC_AIDR	None

5.4.1.10 SSC_PIDR4, Sender Security Peripheral ID 4 Register

Returns byte[4] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR4 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FD0

Bit descriptions

Figure 5-42: MHUS.SSC_SSC_PIDR4 bit assignments

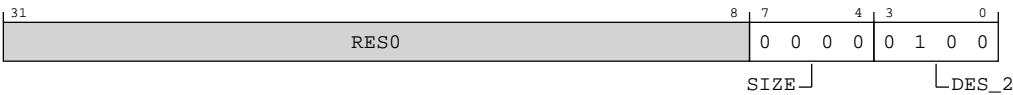


Table 5-86: SSC_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	<div>Size of the component</div> <div>0b0000</div> <div>The size of the component must be identified using a combination of the following registers:</div> <ul style="list-style-type: none">Peripheral ID 0-7 registersComponent ID 0-3 registers<x>_FEAT_SPT0 and <x>_FEAT_SPT1 registers	0b0000
[3:0]	DES_2	<div>JEP106 continuation code</div> <div>0b0100</div>	0b0100

Accessibility

Table 5-87: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FD0	SSC_PIDR4	None

5.4.1.11 SSC_PIDR5, Sender Security Peripheral ID 5 Register

Returns byte[5] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR5 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FD4

Bit descriptions

Figure 5-43: MHUS.SSC_SSC_PIDR5 bit assignments

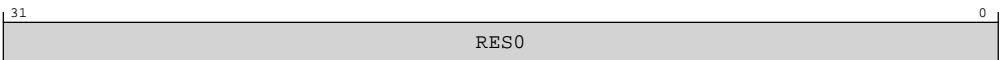


Table 5-88: SSC_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-89: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FD4	SSC_PIDR5	None

5.4.1.12 SSC_PIDR6, Sender Security Peripheral ID 6 Register

Returns byte[6] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR6 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FD8

Bit descriptions

Figure 5-44: MHUS.SSC_SSC_PIDR6 bit assignments

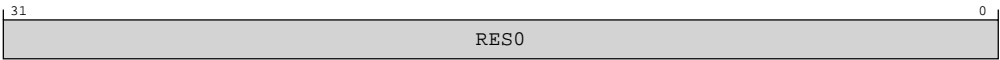


Table 5-90: SSC_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-91: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FD8	SSC_PIDR6	None

5.4.1.13 SSC_PIDR7, Sender Security Peripheral ID 7 Register

Returns byte[7] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR7 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FDC

Bit descriptions

Figure 5-45: MHUS.SSC_SSC_PIDR7 bit assignments

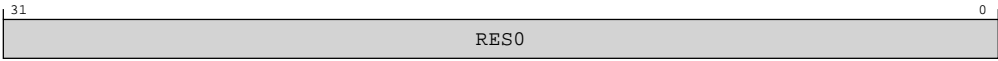


Table 5-92: SSC_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-93: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FDC	SSC_PIDR7	None

5.4.1.14 SSC_PIDR0, Sender Security Peripheral ID 0 Register

Returns byte[0] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FE0

Bit descriptions

Figure 5-46: MHUS.SSC_SSC_PIDR0 bit assignments

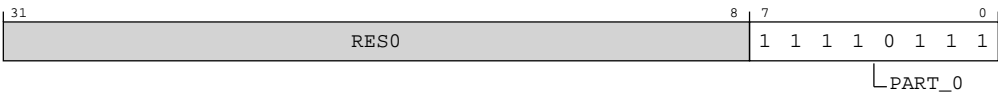


Table 5-94: SSC_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PART_0	Bits 7:0 of the Part ID for the implementation of the MHU 0b111110111	0xF7

Accessibility

Table 5-95: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FE0	SSC_PIDR0	None

5.4.1.15 SSC_PIDR1, Sender Security Peripheral ID 1 Register

Returns byte[1] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR1 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FE4

Bit descriptions

Figure 5-47: MHUS.SSC_SSC_PIDR1 bit assignments

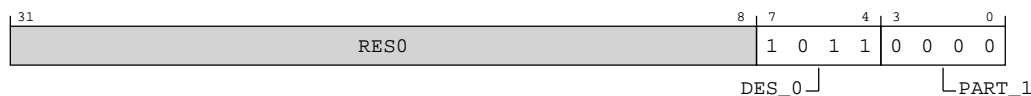


Table 5-96: SSC_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	DES_0	Bits 3:0 of the JEP106 identification code 0b1011	0b1011
[3:0]	PART_1	Bits 11:8 of the Part ID for the implementation of the MHU 0b0000	0b0000

Accessibility

Table 5-97: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FE4	SSC_PIDR1	None

5.4.1.16 SSC_PIDR2, Sender Security Peripheral ID 2 Register

Returns byte[2] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR2 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FE8

Bit descriptions

Figure 5-48: MHUS.SSC_SSC_PIDR2 bit assignments

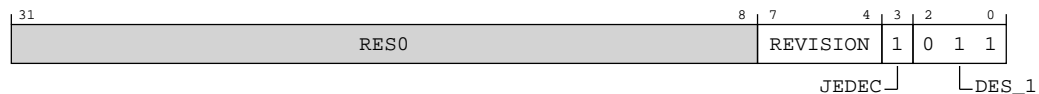


Table 5-98: SSC_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	The REVISION field is an incremental value starting at 0x0 for the first design of a component. The value is increased by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision.	The reset value depends on the product version used. <ul style="list-style-type: none"> 0x0 - r0p0 0x1 - r0p1
[3]	JEDEC	Must be 0b1 to indicate that a JEDEC-assigned value is used. 0b1	0b1
[2:0]	DES_1	JEP106 identification bits [6:4] 0b011	0b011

Accessibility

Table 5-99: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FE8	SSC_PIDR2	None

5.4.1.17 SSC_PIDR3, Sender Security Peripheral ID 3 Register

Returns byte[3] of the peripheral ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_PIDR3 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FEC

Bit descriptions

Figure 5-49: MHUS.SSC_SSC_PIDR3 bit assignments



Table 5-100: SSC_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	The REVAND field indicates minor errata fixes specific to this design, for example metal fixes after implementation. Usually this field is zero. Together with PIDR2.REVISION, PIDR3.REVAND forms the revision number of the component.	0x0

Bits	Name	Description	Reset
[3:0]	CMOD	<p>Customer Modified.</p> <p>If the component is reusable IP, the CMOD field indicates whether the customer has modified the behavior of the component.</p> <p>Arm recommends that the user or debugger reads the documentation for the component to determine the modifications that are made to the component.</p> <p>For any two components with the same values of the Peripheral ID 0-7 and Component ID 0-3 registers</p> <ul style="list-style-type: none"> • If the value of the CMOD fields of both components equals zero, the components are identical • If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have been subjected to the same modifications. • If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same values of the Peripheral ID 0-7 and Component ID 0-3 registers 	0x0

Accessibility

Table 5-101: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FEC	SSC_PIDR3	None

5.4.1.18 SSC_CIDR0, Sender Security Component ID 0 Register

Returns byte[0] of the component ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_CIDR0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FF0

Bit descriptions

Figure 5-50: MHUS.SSC_SSC_CIDR0 bit assignments

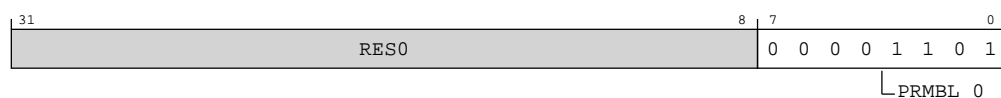


Table 5-102: SSC_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble, segment 0 0b00001101	0x0D

Accessibility

Table 5-103: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FF0	SSC_CIDR0	None

5.4.1.19 SSC_CIDR1, Sender Security Component ID 1 Register

Returns byte[1] of the component ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_CIDR1 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FF4

Bit descriptions

Figure 5-51: MHUS.SSC_SSC_CIDR1 bit assignments



Table 5-104: SSC_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component Class 0b1111 CoreLink, PrimeCell, or system component with no standardized register layout, for backwards compatibility.	0b1111

Bits	Name	Description	Reset
[3:0]	PRMBL_1	Preamble, segment 1 0b0000	0b0000

Accessibility

Table 5-105: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FF4	SSC_CIDR1	None

5.4.1.20 SSC_CIDR2, Sender Security Component ID 2 Register

Returns byte[2] of the component ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_CIDR2 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FF8

Bit descriptions

Figure 5-52: MHUS.SSC_SSC_CIDR2 bit assignments

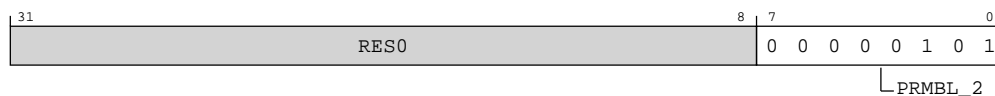


Table 5-106: SSC_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble, segment 2 0b00000101	0x05

Accessibility

Table 5-107: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FF8	SSC_CIDR2	None

5.4.1.21 SSC_CIDR3, Sender Security Component ID 3 Register

Returns byte[3] of the component ID for Sender Security page.

Configurations

This register is present only when TZE is implemented for the MHUS. Otherwise, direct accesses to SSC_CIDR3 are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0x0FFC

Bit descriptions

Figure 5-53: MHUS.SSC_SSC_CIDR3 bit assignments



Table 5-108: SSC_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble, segment 3 0b10110001	0xB1

Accessibility

Table 5-109: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0x0FFC	SSC_CIDR3	None

5.4.1.22 SSC_ACTRL, MHUS Access Control Register

Allows overriding access control in MHUS

Configurations

This register is present only when TZE is implemented for the MHUS and RME is implemented for the MHUS. Otherwise, direct accesses to SSC_ACTRL are RAZ/WI.

Attributes

Width

32

Component

MHUS.SSC

Register offset

0xF000

Bit descriptions

Figure 5-54: MHUS.SSC_SSC_ACTRL bit assignments

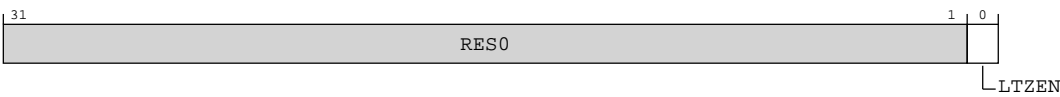


Table 5-110: SSC_ACTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	LTZEN	Allows overriding LEGACY_TZ_EN tie-off 0b0 LEGACY_TZ_EN tie-off is not set. 0b1 LEGACY_TZ_EN tie-off is set. Reverted to TrustZone support. On a MHUS reset, this field resets to the LEGACY_TZ_EN tie-off value of the implementation.	The reset value for this field depends on the MHU configuration.

Accessibility

Table 5-111: Accessibility

Component	Offset	Instance	Range
MHUS.SSC	0xF000	SSC_ACTRL	None

5.4.2 MHU Sender Postbox register summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MHU Sender Postbox (MHUS.PBX) registers.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, see the individual field resets documented on the register description pages or the [Message Handling Unit Architecture version 3.0](#).

Table 5-112: MHUS.PBX register summary

Offset	Name	Type	Reset	Width	Description
0x0000	PBX_BLK_ID	RO	See individual bit resets.	32-bit	Postbox Block Identifier Register
0x0010	PBX_FEAT_SPT0	RO	See individual bit resets.	32-bit	Postbox Feature Support 0 Register
0x0014	PBX_FEAT_SPT1	RO	See individual bit resets.	32-bit	Postbox Feature Support 1 Register
0x0020	PBX_DBCH_CFG0	RO	See individual bit resets.	32-bit	Postbox Doorbell Channel Configuration 0 Register
0x0030	PBX_FFCH_CFG0	RO	See individual bit resets.	32-bit	Postbox FIFO Channel Configuration 0 Register
0x0040	PBX_FCH_CFG0	RO	See individual bit resets.	32-bit	Postbox Fast Channel Configuration 0 Register
0x0100	PBX_CTRL	RW	See individual bit resets.	32-bit	Postbox Control Register
(4 * n) + 0x0400	PBX_DBCH_INT_ST<n>	RO	See individual bit resets.	32-bit	Postbox Doorbell Channel Interrupt Status n Register
(4 * n) + 0x0410	PBX_FFCH_INT_ST<n>	RO	See individual bit resets.	32-bit	Postbox FIFO Channel Interrupt Status n Register
0x0FC8	PBX_IIDR	RO	See individual bit resets.	32-bit	Postbox Implementer Identification Register
0x0FCC	PBX_AIDR	RO	See individual bit resets.	32-bit	Postbox Architecture Identification Register
0x0FD0	PBX_PIDR4	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 4 Register
0x0FD4	PBX_PIDR5	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 5 Register
0x0FD8	PBX_PIDR6	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 6 Register
0x0FDC	PBX_PIDR7	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 7 Register
0x0FE0	PBX_PIDR0	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 0 Register
0x0FE4	PBX_PIDR1	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 1 Register

Offset	Name	Type	Reset	Width	Description
0x0FE8	PBX_PIDR2	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 2 Register
0x0FEC	PBX_PIDR3	RO	See individual bit resets.	32-bit	Postbox Peripheral ID 3 Register
0x0FF0	PBX_CIDR0	RO	See individual bit resets.	32-bit	Postbox Component ID 0 Register
0x0FF4	PBX_CIDR1	RO	See individual bit resets.	32-bit	Postbox Component ID 1 Register
0x0FF8	PBX_CIDR2	RO	See individual bit resets.	32-bit	Postbox Component ID 2 Register
0x0FFC	PBX_CIDR3	RO	See individual bit resets.	32-bit	Postbox Component ID 3 Register
(32 * n) + 0x1000	PDBCW<n>_ST	RO	See individual bit resets.	32-bit	Postbox Doorbell Channel Window <n> Status Register
(32 * n) + 0x100C	PDBCW<n>_SET	WO	See individual bit resets.	32-bit	Postbox Doorbell Channel Window <n> Set Register
(32 * n) + 0x1010	PDBCW<n>_INT_ST	RO	See individual bit resets.	32-bit	Postbox Doorbell Channel Window <n> Interrupt Status Register
(32 * n) + 0x1014	PDBCW<n>_INT_CLR	WO	See individual bit resets.	32-bit	Postbox Doorbell Channel Window <n> Interrupt Clear Register
(32 * n) + 0x1018	PDBCW<n>_INT_EN	RW	See individual bit resets.	32-bit	Postbox Doorbell Channel Window <n> Interrupt Enable Register
(32 * n) + 0x101C	PDBCW<n>_CTRL	RW	See individual bit resets.	32-bit	Postbox Doorbell Channel Window <n> Control Register
(64 * n) + 0x2000	PFFCW<n>_PAY64	RW	See individual bit resets.	64-bit	Postbox FIFO Channel Window <n> Payload Register (64bit access)
(64 * n) + 0x2000	PFFCW<n>_PAY32	RW	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Payload Register (32bit access)
(64 * n) + 0x2004	PFFCW<n>_PAY32	RW	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Payload Register (32bit access)
(64 * n) + 0x2008	PFFCW<n>_FLG	RW	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Flag Register
(64 * n) + 0x2010	PFFCW<n>_INT_ST	RO	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Interrupt Status Register
(64 * n) + 0x2014	PFFCW<n>_INT_CLR	WO	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Interrupt Clear Register
(64 * n) + 0x2018	PFFCW<n>_INT_EN	RW	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Interrupt Enable Register
(64 * n) + 0x2020	PFFCW<n>_CTRL	RW	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Control Register
(64 * n) + 0x2024	PFFCW<n>_ST	RO	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Status Register
(64 * n) + 0x2028	PFFCW<n>_ACK_CNT	RO	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Acknowledge Counter Register
(64 * n) + 0x202C	PFFCW<n>_TIDE	RW	See individual bit resets.	32-bit	Postbox FIFO Channel Window <n> Tidemark Register

Offset	Name	Type	Reset	Width	Description
(4 * n) + 0x3000	PFCW<n>_PAY32	RW	See individual bit resets.	32-bit	Postbox Fast Channel Window <n> Payload 32bit Register
(8 * n) + 0x3000	PFCW<n>_PAY64	RW	See individual bit resets.	64-bit	Postbox Fast Channel Window <n> Payload 64bit Register
0xF000	PBX_FCTRL	RW	See individual bit resets.	32-bit	Postbox Feature Control Register
0xF010	PBX_FIFO_ERRINS	RW	See individual bit resets.	64-bit	Postbox FIFO channel RAM ECC Error Insertion Register

5.4.2.1 PBX_BLK_ID, Postbox Block Identifier Register

Identifies the block as a Postbox.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0000

Bit descriptions

Figure 5-55: MHUS.PBX_PBX_BLK_ID bit assignments

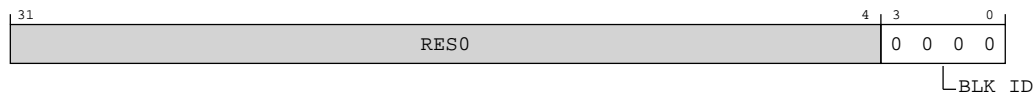


Table 5-113: PBX_BLK_ID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	BLK_ID	Block Identifier Identifies the block as a Postbox 0b0000 Postbox	0b0000

Accessibility

Table 5-114: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0000	PBX_BLK_ID	None

5.4.2.2 PBX_FEAT_SPT0, Postbox Feature Support 0 Register

Returns information on supported MHU features

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0010

Bit descriptions

Figure 5-56: MHUS.PBX_PBX_FEAT_SPT0 bit assignments

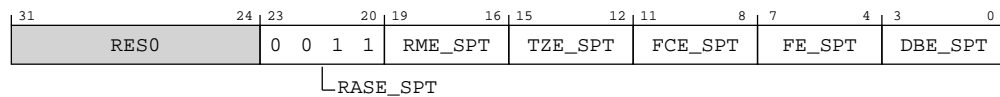


Table 5-115: PBX_FEAT_SPT0 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:20]	RASE_SPT	Reliability, Availability and Serviceability Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0011 MHU implements the RAS extension and follows the recommendations in the "Recommend implementation of RAS using Arm RAS extensions" section of Message Handling Unit Architecture version 3.0 .	0b0011

Bits	Name	Description	Reset
[19:16]	RME_SPT	<p>Realm Management Extension Support</p> <p>The value of this field depends on the implementation of the MHU and an optional reset time sampled input LEGACY_TZ_EN. The field can take one of the following values:</p> <p>0b0000 MHU does not implement the Realm Management extension</p> <p>0b0001 MHU implements Realm Management extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p> <p>When RME is implemented, for the MHU component, there can be a LEGACY_TZ_EN tie-off signal present on this MHU component.</p> <p>The value of the LEGACY_TZ_EN tie-off signal is sampled at reset of the MHU component which the tie-off signal is associated with.</p> <p>When the sampled value of the tie-off signal is 0b1 the value of this field is always 0x0, otherwise the value of this field is dependent on whether RME is implemented for this MHU component.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>
[15:12]	TZE_SPT	<p>TrustZone Extension Support</p> <p>The value of this field depends on the implementation of the MHU and can take one of the following values:</p> <p>0b0000 MHU does not implement the TrustZone extension</p> <p>0b0001 MHU implements TrustZone extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>

Bits	Name	Description	Reset
[11:8]	FCE_SPT	Fast Channel Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Fast Channel extension 0b0001 MHU implements Fast Channel extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[7:4]	FE_SPT	FIFO Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the FIFO extension 0b0001 MHU implements FIFO extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[3:0]	DBE_SPT	Doorbell Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Doorbell extension 0b0001 MHU implements Doorbell extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.

Accessibility

Table 5-116: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0010	PBX_FEAT_SPT0	None

5.4.2.3 PBX_FEAT_SPT1, Postbox Feature Support 1 Register

Returns information on supported MHU features

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset
0x0014

Bit descriptions

Figure 5-57: MHUS.PBX_PBX_FEAT_SPT1 bit assignments

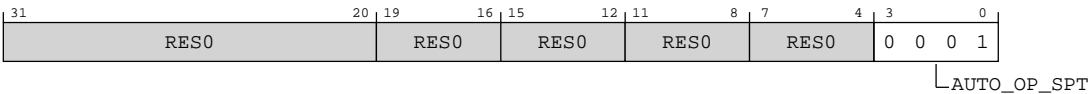


Table 5-117: PBX_FEAT_SPT1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	AUTO_OP_SPT	Auto Op Protocol support For more information about the Auto Op protocol, see the Message Handling Unit Architecture version 3.0 . The value of this field is set for MHU-320AE: 0b0001 Auto Op(Full) is implemented	0b0001

Accessibility

Table 5-118: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0014	PBX_FEAT_SPT1	None

5.4.2.4 PBX_DBCH_CFG0, Postbox Doorbell Channel Configuration 0 Register

Returns doorbell channel configuration information

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PBX_DBCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0020

Bit descriptions

Figure 5-58: MHUS.PBX_PBX_DBCH_CFG0 bit assignments

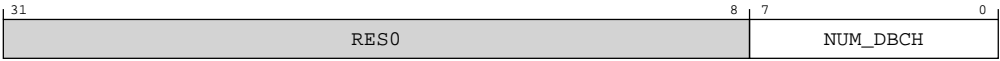


Table 5-119: PBX_DBCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	NUM_DBCH	Number of Doorbell Channels 0b00000000..0b01111111 Number of DBCH is N+1, where N is the value of this field	8{x}

Accessibility

Table 5-120: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0020	PBX_DBCH_CFG0	None

5.4.2.5 PBX_FFCH_CFG0, Postbox FIFO Channel Configuration 0 Register

Returns FIFO channel configuration information

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PBX_FFCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0030

Bit descriptions

Figure 5-59: MHUS.PBX_PBX_FFCH_CFG0 bit assignments

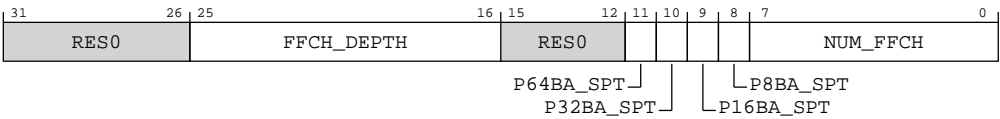


Table 5-121: PBX_FFCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25:16]	FFCH_DEPTH	FIFO Channel Depth 0b0000000000..0b1111111111 FIFO depth is N+1 bytes, where N is the value of this field	10{x}
[15:12]	RES0	Reserved	RES0
[11]	P64BA_SPT	Postbox 64bit Access Support Whether the implementation of the MHU supports 64bit accesses to the PFFCW<n>_PAY register 0b0 64bit accesses are not supported 0b1 64bit accesses are supported Accesses must be aligned to an 64bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.
[10]	P32BA_SPT	Postbox 32bit Access Support Whether the implementation of the MHU supports 32bit accesses to the PFFCW<n>_PAY register 0b0 32bit accesses are not supported 0b1 32bit accesses are supported Accesses must be aligned to an 32bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.
[9]	P16BA_SPT	Postbox 16bit Access Support Whether the implementation of the MHU supports 16bit accesses to the PFFCW<n>_PAY register 0b0 16bit accesses are not supported 0b1 16bit accesses are supported Accesses must be aligned to an 16bit boundary	0x0

Bits	Name	Description	Reset
[8]	P8BA_SPT	Postbox 8bit Access Support Whether the implementation of the MHU supports 8bit accesses to the PFFCW<n>_PAY register 0b0 8bit accesses are not supported 0b1 8bit accesses are supported Accesses must be aligned to an 8bit boundary	0x0
[7:0]	NUM_FFCH	Number of FIFO Channels The number of FIFO Channels in the Postbox. 0b00000000..0b00111111 Number of FIFO is N+1, where N is the value of this field	8 {x}

Accessibility

Table 5-122: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0030	PBX_FFCH_CFG0	None

5.4.2.6 PBX_FCH_CFG0, Postbox Fast Channel Configuration 0 Register

Returns fast channel configuration information

Configurations

This register is present only when FCE is implemented. Otherwise, direct accesses to PBX_FCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0040

Bit descriptions

Figure 5-60: MHUS.PBX_PBX_FCH_CFG0 bit assignments

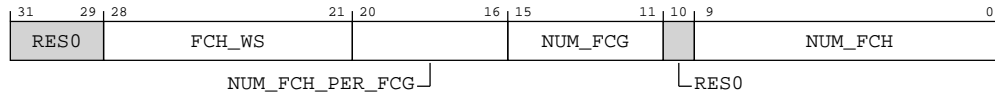


Table 5-123: PBX_FCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:29]	RES0	Reserved	RES0
[28:21]	FCH_WS	<p>Fast Channel Word Size</p> <p>Number of bits each Fast Channel implements.</p> <p>The value must be the same as MBX_FCH_CFG0.FCH_WS.</p> <p>0b00100000 Fast Channel word size is 32-bits</p> <p>0b01000000 Fast Channel word size is 64-bits</p>	The reset values can be the following: 0b00100000, 0b01000000, respective to the value.
[20:16]	NUM_FCH_PER_FCG	<p>Number of Fast Channels per Fast Channel Group for the Postbox</p> <p>0b000000..0b11111 Number of Fast Channels per Fast Channel Group is N+1, where N is the value of this field</p> <p>The number of Fast Channels in the last Fast Channel Group can be less than the value in this field if the value of NUM_FCH is not a multiple of the number of Fast Channels per Fast Channel Group.</p> <p>All other values are Reserved</p>	The reset value for this field depends on the MHU configuration.
[15:11]	NUM_FCG	<p>Number of Fast Channel Groups for the Postbox</p> <p>0b000000..0b11111 Number of Fast Channel Groups is N+1, where N is the value of this field</p> <p>The legal values for this field are 0-31</p>	The reset value for this field depends on the MHU configuration.
[10]	RES0	Reserved	RES0
[9:0]	NUM_FCH	<p>Number of Fast Channels in the Postbox</p> <p>FCH_WS == 0x40 0b000000000000..0b0111111111 Number of FCH is N+1, where N is the value of this field</p> <p>FCH_WS == 0x20 0b000000000000..0b1111111111 Number of FCH is N+1, where N is the value of this field</p>	The reset value for this field depends on the MHU configuration.

Accessibility

Table 5-124: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0040	PBX_FCH_CFG0	None

5.4.2.7 PBX_CTRL, Postbox Control Register

This register contains control bits for the postbox

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0100

Bit descriptions

Figure 5-61: MHUS.PBX_PBX_CTRL bit assignments

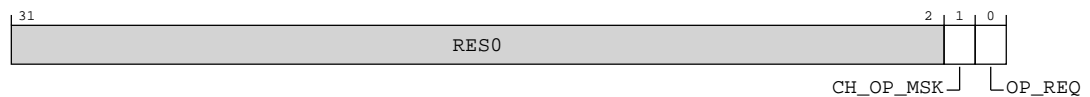


Table 5-125: PBX_CTRL bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	CH_OP_MSK	<p>Channel Operational Mask</p> <p>Controls whether Channels need to be idle to allow a controlled entry into a non-operational state</p> <p>0b0</p> <p>Channels must be idle to enter a non-operational state</p> <p>0b1</p> <p>Channels status is ignored when considering entry into a non-operational state</p> <p>This field has no effect on entry into a non-operation state in an uncontrolled manner</p>	0b0

Bits	Name	Description	Reset
[0]	OP_REQ	Operational Request 0b0 Postbox is not requested to remain in the operational state 0b1 Postbox is requested to remain in the operational state	0b0

Accessibility

Table 5-126: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0100	PBX_CTRL	None

5.4.2.8 PBX_DBCH_INT_ST<n>, Postbox Doorbell Channel Interrupt Status n Register, n = 0 - 3

Indicates whether there is an interrupt outstanding for a Doorbell Channel

PBX_DBCH_INT_ST0 has status fields for Doorbell Channels 0 to 31

PBX_DBCH_INT_ST1 has status fields for Doorbell Channels 32 to 63

PBX_DBCH_INT_ST2 has status fields for Doorbell Channels 64 to 95

PBX_DBCH_INT_ST3 has status fields for Doorbell Channels 96 to 127

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PBX_DBCH_INT_ST<n> are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(4 * n) + 0x0400

Bit descriptions

Figure 5-62: MHUS.PBX_PBX_DBCH_INT_ST<n> bit assignments

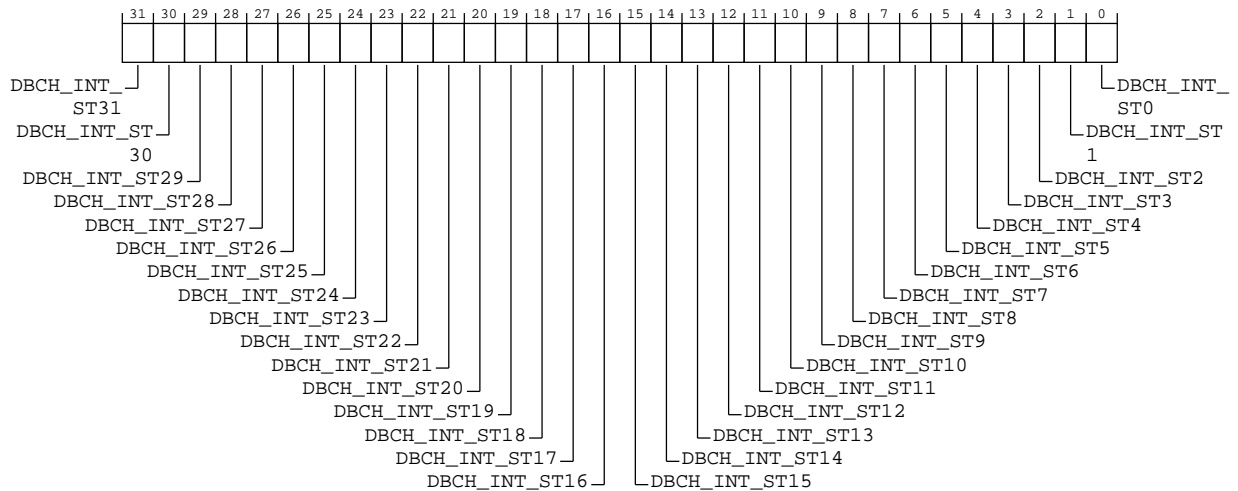


Table 5-127: PBX_DBCH_INT_ST<n> bit descriptions

Bits	Name	Description	Reset
[31:0]	DBCH_INT_ST<x>	<p>Doorbell Channel Interrupt Status</p> <p>Each bit indicates whether there is an interrupt outstanding for Doorbell Channel</p> <p>0b0</p> <p>No interrupt outstanding for the Doorbell Channel or the Channel is not configured to factor into the Postbox Combined interrupt.</p> <p>0b1</p> <p>Interrupt outstanding for the Doorbell Channel and the Channel is configured to factor into the Postbox Combined interrupt.</p> <p>Any fields which are not assigned to a Channel are Reserved and treated as RES0</p>	0x00000000

Accessibility

Table 5-128: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(4 * n) + 0x0400	PBX_DBCH_INT_ST<n>	None

5.4.2.9 PBX_FFCH_INT_ST<n>, Postbox FIFO Channel Interrupt Status n Register, n = 0 - 1

Indicates whether there is an interrupt outstanding for the FIFO Channel.

PBX_FFCH_INT_ST0 has status fields for FIFO Channels 0 to 31

PBX_FFCH_INT_ST1 has status fields for FIFO Channels 32 to 63

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PBX_FFCH_INT_ST<n> are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(4 * n) + 0x0410

Bit descriptions

Figure 5-63: MHUS.PBX_PBX_FFCH_INT_ST<n> bit assignments

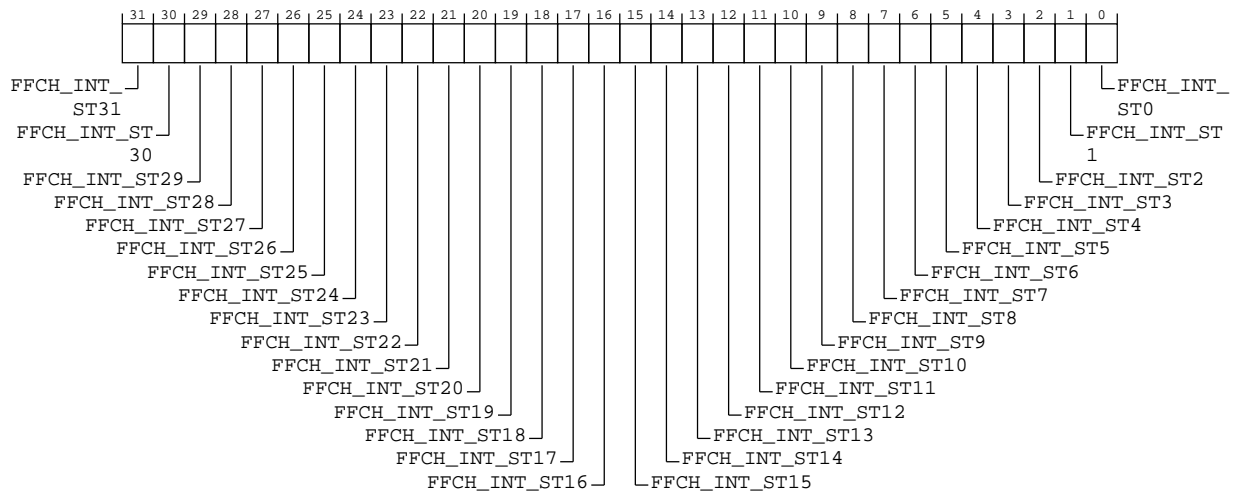


Table 5-129: PBX_FFCH_INT_ST<n> bit descriptions

Bits	Name	Description	Reset
[31:0]	FFCH_INT_ST<x>	<p>FIFO Channel Interrupt Status</p> <p>Each bit indicates whether there is an interrupt outstanding for the FIFO Channel</p> <p>0b0</p> <p>No interrupt outstanding for the FIFO Channel or the Channel is not configured to factor into the Postbox Combined interrupt.</p> <p>0b1</p> <p>Interrupt outstanding for the FIFO Channel and the Channel is configured to factor into the Postbox Combined interrupt.</p> <p>Any fields which are not assigned to a Channel are Reserved and treated as RAZ/WI</p>	0x00000000

Accessibility

Table 5-130: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	$(4 * n) + 0x0410$	PBX_FFCH_INT_ST<n>	None

5.4.2.10 PBX_IIDR, Postbox Implementer Identification Register

This field provides information on the Implementer of the MHU

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FC8

Bit descriptions

Figure 5-64: MHUS.PBX_PBX_IIDR bit assignments

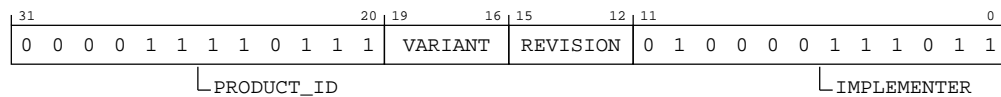


Table 5-131: PBX_IIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	PRODUCT_ID	Product ID of the MHU implementation 0b000011110111	0x0F7
[19:16]	VARIANT	Variant or Major revision of the MHU implementation	The reset value depends on the product version used. <ul style="list-style-type: none"> 0x0 - r0
[15:12]	REVISION	Revision or minor version of the MHU implementation	The reset value depends on the product version used. <ul style="list-style-type: none"> 0x0 - p0 0x1 - p1

Bits	Name	Description	Reset
[11:0]	IMPLEMENTER	Implementer ID Contains the JEP106 identification information as follows: <ul style="list-style-type: none">11:8 - JEP106 continuation code of implementer7 - Always 06:0 - JEP106 identity code of implementer 0b010000111011	0x43B

Accessibility

Table 5-132: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FC8	PBX_IIDR	None

5.4.2.11 PBX_AIDR, Postbox Architecture Identification Register

Provides information on the implemented MHU architecture

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FCC

Bit descriptions

Figure 5-65: MHUS.PBX_PBX_AIDR bit assignments

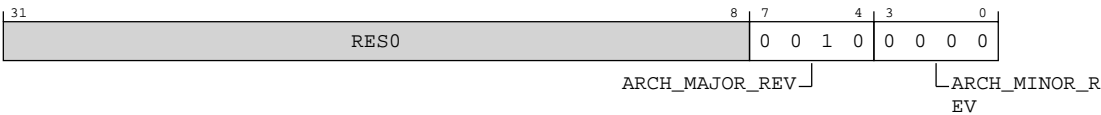


Table 5-133: PBX_AIDR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	ARCH_MAJOR_REV	MHU Architecture Major Revision 0b0010 MHUv3 All other values are Reserved	0b0010
[3:0]	ARCH_MINOR_REV	MHU Architecture Minor Revision 0b0000 Minor revision 0 of the major architecture All other values are Reserved	0b0000

Accessibility

Table 5-134: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FCC	PBX_AIDR	None

5.4.2.12 PBX_PIDR4, Postbox Peripheral ID 4 Register

Returns byte[4] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FD0

Bit descriptions

Figure 5-66: MHUS.PBX_PBX_PIDR4 bit assignments

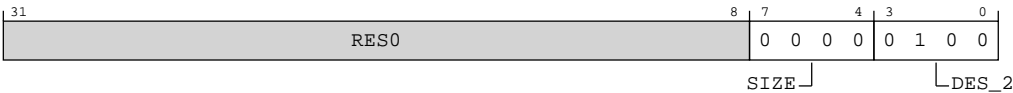


Table 5-135: PBX_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	SIZE	Size of the component 0b0000 The size of the component must be identified using a combination of the following registers: <ul style="list-style-type: none"> Peripheral ID 0-7 registers Component ID 0-3 registers <x>_FEAT_SPT0 and <x>_FEAT_SPT1 registers 	0b0000
[3:0]	DES_2	JEP106 continuation code 0b0100	0b0100

Accessibility

Table 5-136: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FD0	PBX_PIDR4	None

5.4.2.13 PBX_PIDR5, Postbox Peripheral ID 5 Register

Returns byte[5] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FD4

Bit descriptions

Figure 5-67: MHUS.PBX_PBX_PIDR5 bit assignments



Table 5-137: PBX_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-138: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FD4	PBX_PIDR5	None

5.4.2.14 PBX_PIDR6, Postbox Peripheral ID 6 Register

Returns byte[6] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FD8

Bit descriptions

Figure 5-68: MHUS.PBX_PBX_PIDR6 bit assignments

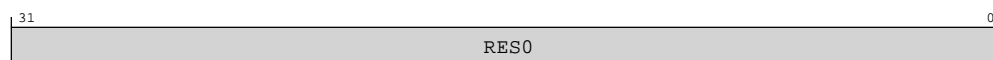


Table 5-139: PBX_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-140: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FD8	PBX_PIDR6	None

5.4.2.15 PBX_PIDR7, Postbox Peripheral ID 7 Register

Returns byte[7] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FDC

Bit descriptions

Figure 5-69: MHUS.PBX_PBX_PIDR7 bit assignments

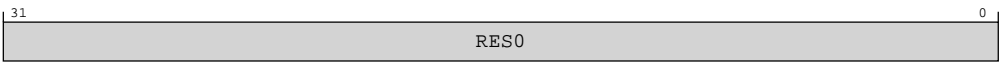


Table 5-141: PBX_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-142: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FDC	PBX_PIDR7	None

5.4.2.16 PBX_PIDR0, Postbox Peripheral ID 0 Register

Returns byte[0] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
MHUS.PBX

Register offset
0x0FE0

Bit descriptions

Figure 5-70: MHUS.PBX_PBX_PIDR0 bit assignments

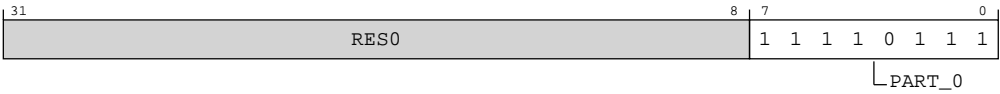


Table 5-143: PBX_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Bits 7:0 of the Part ID for the implementation of the MHU 0b11110111	0xF7

Accessibility

Table 5-144: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FE0	PBX_PIDR0	None

5.4.2.17 PBX_PIDR1, Postbox Peripheral ID 1 Register

Returns byte[1] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUS.PBX

Register offset
0x0FE4

Bit descriptions

Figure 5-71: MHUS.PBX_PBX_PIDR1 bit assignments

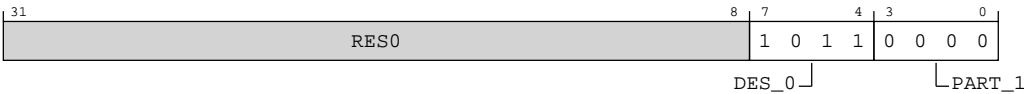


Table 5-145: PBX_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	DES_0	Bits 3:0 of the JEP106 identification code 0b1011	0b1011
[3:0]	PART_1	Bits 11:8 of the Part ID for the implementation of the MHU 0b0000	0b0000

Accessibility

Table 5-146: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FE4	PBX_PIDR1	None

5.4.2.18 PBX_PIDR2, Postbox Peripheral ID 2 Register

Returns byte[2] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FE8

Bit descriptions

Figure 5-72: MHUS.PBX_PBX_PIDR2 bit assignments

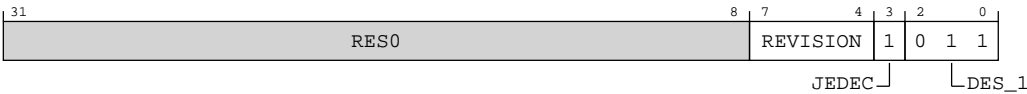


Table 5-147: PBX_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	The REVISION field is an incremental value starting at 0x0 for the first design of a component. The value is increased by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision.	The reset value depends on the product version used. <ul style="list-style-type: none">0x0 - r0p00x1 - r0p1
[3]	JEDEC	Must be 0b1 to indicate that a JEDEC-assigned value is used. 0b1	0b1
[2:0]	DES_1	JEP106 identification bits [6:4] 0b011	0b011

Accessibility

Table 5-148: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FE8	PBX_PIDR2	None

5.4.2.19 PBX_PIDR3, Postbox Peripheral ID 3 Register

Returns byte[3] of the peripheral ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FEC

Bit descriptions

Figure 5-73: MHUS.PBX_PBX_PIDR3 bit assignments



Table 5-149: PBX_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	The REVAND field indicates minor errata fixes specific to this design, for example metal fixes after implementation. Usually this field is zero. Together with PIDR2.REVISION, PIDR3.REVAND forms the revision number of the component.	0x0
[3:0]	CMOD	Customer Modified. If the component is reusable IP, the CMOD field indicates whether the customer has modified the behavior of the component. Arm recommends that the user or debugger reads the documentation for the component to determine the modifications that are made to the component. For any two components with the same values of the Peripheral ID 0-7 and Component ID 0-3 registers <ul style="list-style-type: none"> If the value of the CMOD fields of both components equals zero, the components are identical If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have been subjected to the same modifications. If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same values of the Peripheral ID 0-7 and Component ID 0-3 registers 	0x0

Accessibility

Table 5-150: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FEC	PBX_PIDR3	None

5.4.2.20 PBX_CIDR0, Postbox Component ID 0 Register

Returns byte[0] of the component ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
MHUS.PBX

Register offset
0x0FF0

Bit descriptions

Figure 5-74: MHUS.PBX_PBX_CIDR0 bit assignments



Table 5-151: PBX_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble, segment 0 0b00001101	0x0D

Accessibility

Table 5-152: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FF0	PBX_CIDR0	None

5.4.2.21 PBX_CIDR1, Postbox Component ID 1 Register

Returns byte[1] of the component ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUS.PBX

Register offset
0x0FF4

Bit descriptions

Figure 5-75: MHUS.PBX_PBX_CIDR1 bit assignments

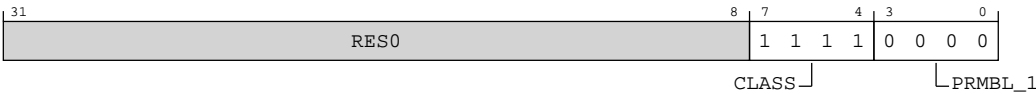


Table 5-153: PBX_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component Class 0b1111 CoreLink, PrimeCell, or system component with no standardized register layout, for backwards compatibility.	0b1111
[3:0]	PRMBL_1	Preamble, segment 1 0b0000	0b0000

Accessibility

Table 5-154: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FF4	PBX_CIDR1	None

5.4.2.22 PBX_CIDR2, Postbox Component ID 2 Register

Returns byte[2] of the component ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FF8

Bit descriptions

Figure 5-76: MHUS.PBX_PBX_CIDR2 bit assignments

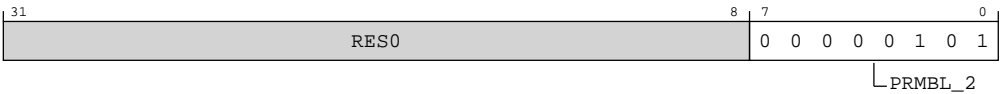


Table 5-155: PBX_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble, segment 2 0b00000101	0x05

Accessibility

Table 5-156: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FF8	PBX_CIDR2	None

5.4.2.23 PBX_CIDR3, Postbox Component ID 3 Register

Returns byte[3] of the component ID for Postbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0x0FFC

Bit descriptions

Figure 5-77: MHUS.PBX_PBX_CIDR3 bit assignments

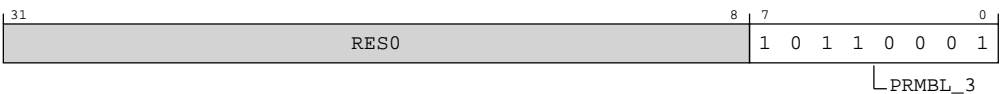


Table 5-157: PBX_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble, segment 3 0b10110001	0xB1

Accessibility

Table 5-158: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0x0FFC	PBX_CIDR3	None

5.4.2.24 PDBCW<n>_ST, Postbox Doorbell Channel Window <n> Status Register, n = 0 - 127

Returns doorbell channel flags

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PDBCW<n>_ST are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(32 * n) + 0x1000

Bit descriptions

Figure 5-78: MHUS.PBX_PDBCW<n>_ST bit assignments

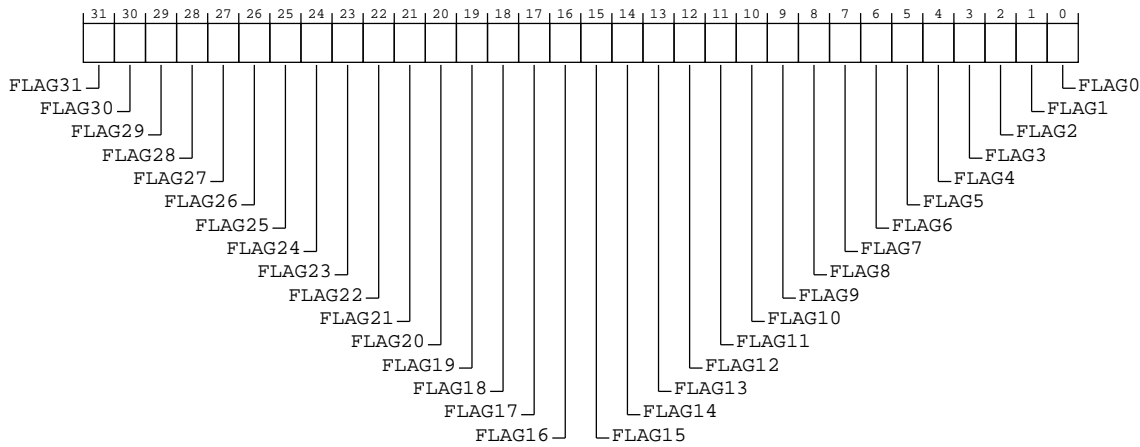


Table 5-159: PDBCW<n>_ST bit descriptions

Bits	Name	Description	Reset
[31:0]	FLAG<x>	Indicates the status of Flag bit <x> of the DBCH 0b0 Flag<x> bit is not set 0b1 Flag<x> bit is set	0x00000000

Accessibility

Table 5-160: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(32 * n) + 0x1000	PDBCW<n>_ST	None

5.4.2.25 PDBCW<n>_SET, Postbox Doorbell Channel Window <n> Set Register, n = 0 - 127

Allows setting doorbell channel flags

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PDBCW<n>_SET are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

$(32 * n) + 0x100C$

Bit descriptions

Figure 5-79: MHUS.PBX_PDBCW<n>_SET bit assignments

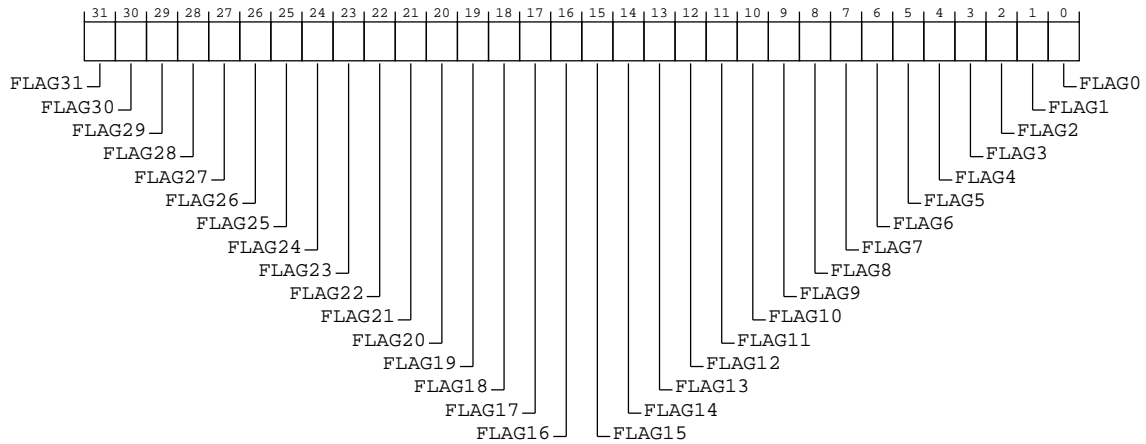


Table 5-161: PDBCW<n>_SET bit descriptions

Bits	Name	Description	Reset
[31:0]	FLAG<x>	Writes of 1 cause the associated bit in the PDBCW<n>_ST registers to be set to 1. Writing 0 has no effect on the value of PDBCW<n>_ST. 0b0 No effect 0b1 Sets the associated bit in the PDBCW<n>_ST register to 1 Field always reads as 0	32 {x}

Accessibility

Table 5-162: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	$(32 * n) + 0x100C$	PDBCW<n>_SET	None

5.4.2.26 PDBCW<n>_INT_ST, Postbox Doorbell Channel Window <n> Interrupt Status Register, n = 0 - 127

Returns doorbell channel interrupt status

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PDBCW<n>_INT_ST are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(32 * n) + 0x1010

Bit descriptions

Shows the status of the events of the DBCH

Figure 5-80: MHUS.PBX_PDBCW<n>_INT_ST bit assignments

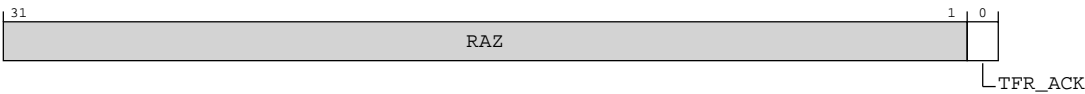


Table 5-163: PDBCW<n>_INT_ST bit descriptions

Bits	Name	Description	Reset
[31:1]	RAZ	Reserved	RAZ
[0]	TFR_ACK	0b0 No Transfer Acknowledge event 0b1 Transfer Acknowledge event	0b0

Accessibility

Table 5-164: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(32 * n) + 0x1010	PDBCW<n>_INT_ST	None

5.4.2.27 PDBCW<n>_INT_CLR, Postbox Doorbell Channel Window <n> Interrupt Clear Register, n = 0 - 127

Register for clearing doorbell channel interrupt status

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PDBCW<n>_INT_CLR are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(32 * n) + 0x1014

Bit descriptions

Clears the status register of the DBCH

Figure 5-81: MHUS.PBX_PDBCW<n>_INT_CLR bit assignments

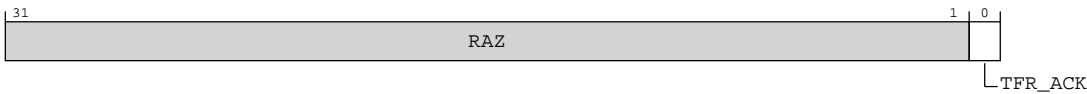


Table 5-165: PDBCW<n>_INT_CLR bit descriptions

Bits	Name	Description	Reset
[31:1]	RAZ	Reserved	RAZ
[0]	TFR_ACK	0b0 No effect 0b1 Clears the Transfer Acknowledge field in the PDBCW<n>_INT_ST register	0b0

Accessibility

Table 5-166: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(32 * n) + 0x1014	PDBCW<n>_INT_CLR	None

5.4.2.28 PDBCW<n>_INT_EN, Postbox Doorbell Channel Window <n> Interrupt Enable Register, n = 0 - 127

Register for configuring doorbell channel interrupt enables

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PDBCW<n>_INT_EN are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(32 * n) + 0x1018

Bit descriptions

Enables which events generate an interrupts

Figure 5-82: MHUS.PBX_PDBCW<n>_INT_EN bit assignments

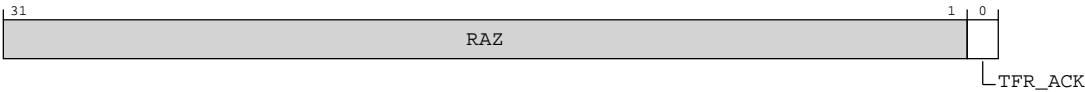


Table 5-167: PDBCW<n>_INT_EN bit descriptions

Bits	Name	Description	Reset
[31:1]	RAZ	Reserved	RAZ
[0]	TFR_ACK	0b0 Transfer Acknowledge events do not generate interrupts 0b1 Transfer Acknowledge events do generate interrupts	0b0

Accessibility

Table 5-168: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(32 * n) + 0x1018	PDBCW<n>_INT_EN	None

5.4.2.29 PDBCW<n>_CTRL, Postbox Doorbell Channel Window <n> Control Register, n = 0 - 127

This register contains control bits for doorbell channels

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to PDBCW<n>_CTRL are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(32 * n) + 0x101C

Bit descriptions

Figure 5-83: MHUS.PBX_PDBCW<n>_CTRL bit assignments

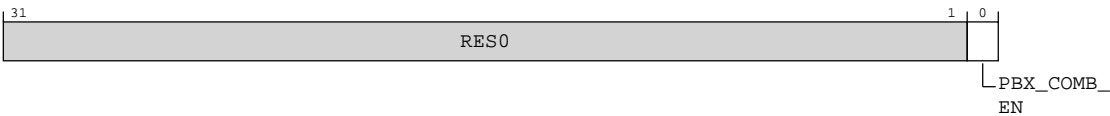


Table 5-169: PDBCW<n>_CTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	PBX_COMB_EN	Controls whether events from this Doorbell Channel contribute to the Postbox Combined interrupt 0b0 Doorbell Channel does not contribute to the Postbox Combined interrupt 0b1 Doorbell Channel contributes to the Postbox Combined interrupt	0b1

Accessibility

Table 5-170: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(32 * n) + 0x101C	PDBCW<n>_CTRL	None

5.4.2.30 PFFCW<n>_PAY64, Postbox FIFO Channel Window <n> Payload Register (64bit access), n = 0 - 63

A 64bit access to the PFFCW<n>_PAY register. The bit descriptions for this register depend on whether the access is a write or a read.

An access must be aligned to any 64bit boundary in the PFFCW_PAY register, otherwise it is an unsupported access. It is implementation-defined whether an unsupported access is treated as RAZ/WI or modified to be an aligned access.

- 64bit accesses are only supported if PBX_FFCH_CFG0.P64BA_SPT is set to 0b1. .
- If PBX_FFCH_CFG0.P64BA_SPT is set to 0b0, 64bit accesses are not supported. It is implementation-defined whether unsupported accesses are treated as RAZ/WI or modified to a supported size.

The PFFCW<n>_PAY register occupies offsets 0x00-0x07 in the Postbox FIFO Channel Window <n>.

A write access pushes the eight bytes that are written to this offset onto the FIFO, if the FIFO has at least eight byte of free space.

Read accesses return the number of bytes free space in the FIFO and whether the previous push operation generated an error or not.

Configurations

This register is present only when FE is implemented and 64bit accesses are supported. Otherwise, direct accesses to PFFCW<n>_PAY64 are RAZ/WI.

Attributes

Width

64

Component

MHUS.PBX

Register offsets

$(64 * n) + 0x2000$

Bit descriptions

When the access is a 64bit write that is aligned to a 64bit boundary, and the MHU implements 64bit accesses to the PFFCW<n>_PAY register, the register has the following bit assignments.

Figure 5-84: PFFCW<n>_PAY64_PFFCW<n>_PAY64 write bit assignments

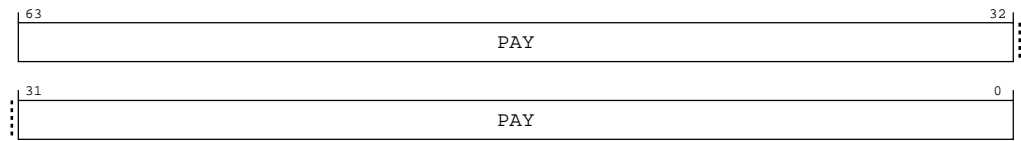


Table 5-171: PFFCW<n>_PAY64 write bit descriptions

Bits	Name	Description	Reset
[63:0]	PAY	<p>Payload to push onto FIFO</p> <p>Causes the written bytes of data to be pushed onto the FIFO, if there is at least eight bytes of free space in the FIFO and sets the PFFCW<n>_ST.PPE field to be set to 0b0.</p> <p>If there is less than eight bytes of free space in the FIFO, then no bytes are pushed onto the FIFO and the PFFCW<n>_ST.PPE field is set to 0b1</p> <p>The value written to this field has no effect on the operation of the FIFO.</p> <p>The order in which bytes are pushed onto the FIFO depends on the value of the PFFCW<n>_CTRL.MSBF field as follows:</p> <ul style="list-style-type: none">0b0 - Bytes are pushed onto the FIFO starting with the LSB0b1 - Bytes are pushed onto the FIFO starting with the MSB <p>The MHU is a little endian device and considers the LSB to be the byte which is written to the lowest offsets accessed by the write.</p> <p>SOT flag is always associated with the first byte pushed onto the FIFO and EOT and ACK flags are associated with the last byte pushed onto the FIFO</p> <p>If the Transfer Delineation Mode for the FIFO is set to Partial or Auto Flag then on a write which sets PFFCW<n>_ST.PPE field to 0b0, the values of data flags in the PFFCW<n>_FLG register are updated.</p>	64 { x }

When the access is a 64bit write that is aligned to a 64bit boundary, and the MHU implements 64bit accesses to the PFFCW<n>_PAY register, the register has the following bit assignments.

Figure 5-85: PFFCW<n>_PAY64_PFFCW<n>_PAY64 read bit assignments

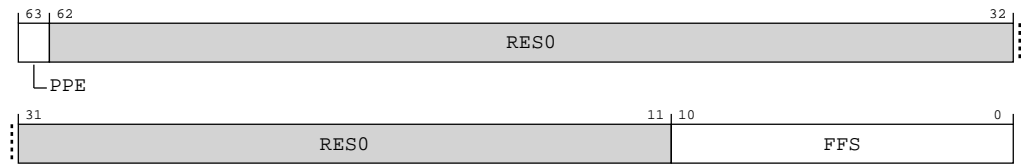


Table 5-172: PFFCW<n>_PAY64 read bit descriptions

Bits	Name	Description	Reset
[63]	PPE	<p>Previous Push Error.</p> <p>Indicates whether a previous push to the FIFO caused an error.</p> <p>An error occurs due to the FIFO not having enough space to push all the bytes provided in the last write to the PFFCW_PAY register.</p> <p>0b0 No error has occurred on the last push operation</p> <p>0b1 An error has occurred on the last push operation</p>	0b0
[62:11]	RES0	Reserved	RES0
[10:0]	FFS	<p>FIFO Free Space.</p> <p>0b000000000000 No free bytes in the FIFO</p> <p>The maximum value returned is never greater than the FIFO Depth.</p>	11 {x}

Accessibility

Table 5-173: Accessibility

Component	Offset	Range
PFFCW<n>_PAY64	[(64 * n) + 0x2000]	63:0

5.4.2.31 PFFCW<n>_PAY32, Postbox FIFO Channel Window <n> Payload Register (32bit access), n = 0 - 63

A 32bit access to the PFFCW<n>_PAY register. The bit descriptions for this register depend on whether the access is a write or a read.

An access must be aligned to any 32bit boundary within the PFFCW_PAY register, otherwise it is an unsupported access. It is implementation-defined whether the access is treated as RAZ/WI or modified to be an aligned access.

- 32bit accesses are only supported if PBX_FFCH_CFG0.P32BA_SPT is set to 0b1.
- If PBX_FFCH_CFG0.P32BA_SPT is set to 0b0, 32bit accesses are not supported. It is implementation-defined whether an unsupported access is treated as RAZ/WI or modified to a supported size.

The number of PFFCW<n>_PAY32 registers depends on the PBX_FFCH_CFG0.P64BA_SPT register field.

- If P64BA_SPT is set to 1, there are two PFFCW<n>_PAY32 registers, which occupy offsets 0x00-0x07 in the Postbox FIFO Channel Window <n>.
- If P64BA_SPT is set to 0, the PFFCW<n>_PAY32 register occupies offsets 0x00-0x04 and offsets 0x05 - 0x07 are reserved, within the Postbox FIFO Channel Window <n>.

A write access pushes the four bytes that are written to this offset onto the FIFO, if the FIFO has at least four byte of free space.

Read accesses return the number of bytes free space in the FIFO and whether the previous push operation generated an error or not.

Configurations

This register is present only when FE is implemented and 32bit accesses are supported. Otherwise, direct accesses to PFFCW<n>_PAY32 are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offsets (4)

$(64 * n) + 0x2000, (64 * n) + 0x2004$

Bit descriptions

When the access is a 32bit write that is aligned to an 32bit boundary, and the MHU implements 32bit accesses to the PFFCW<n>_PAY register, the register has the following bit assignments.

Figure 5-86: PFFCW<n>_PAY32_PFFCW<n>_PAY32 write bit assignments



Table 5-174: PFFCW<n>_PAY32 write bit descriptions

Bits	Name	Description	Reset
[31:0]	PAY	<p>Payload to push onto FIFO</p> <p>Causes the written bytes of data to be pushed onto the FIFO, if there is at least four bytes of free space in the FIFO and sets the PFFCW<n>_ST.PPE field to be set to 0b0.</p> <p>If there is less than four bytes of free space in the FIFO, then no bytes are pushed onto the FIFO and the PFFCW<n>_ST.PPE field is set to 0b1</p> <p>The value written to this field has no effect on the operation of the FIFO.</p> <p>The order in which bytes are pushed onto the FIFO depends on the value of the PFFCW<n>_CTRL.MSBF field as follows:</p> <ul style="list-style-type: none"> 0b0 - Bytes are pushed onto the FIFO starting with the LSB 0b1 - Bytes are pushed onto the FIFO starting with the MSB <p>The MHU is a little endian device and considers the LSB to be the byte which is written to the lowest offsets accessed by the write.</p> <p>SOT flag is always associated with the first byte pushed onto the FIFO and EOT and ACK flags are associated with the last byte pushed onto the FIFO</p> <p>If the Transfer Delineation Mode for the FIFO is set to Partial or Auto Flag then on a write which sets PFFCW<n>_ST.PPE field to 0b0, the values of data flags in the PFFCW<n>_FLG register are updated.</p>	32 {x}

When the access is a 32bit read that is aligned to an 32bit boundary, and the MHU implements 32bit accesses to the PFFCW<n>_PAY register, the register has the following bit assignments.

Figure 5-87: PFFCW<n>_PAY32_PFFCW<n>_PAY32 read bit assignments

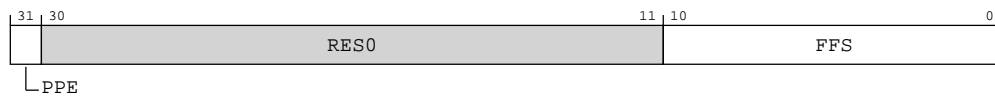


Table 5-175: PFFCW<n>_PAY32 read bit descriptions

Bits	Name	Description	Reset
[31]	PPE	<p>Previous Push Error</p> <p>Indicates whether a previous push to the FIFO caused an error</p> <p>An error occurs due to the FIFO not having enough space to push all the bytes provided in the last write to the PFFCW_PAY register.</p> <p>0b0</p> <p>No error has occurred on the last push operation</p> <p>0b1</p> <p>An error has occurred on the last push operation</p>	–
[30:11]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[10:0]	FFS	FIFO Free Space Indicates the number of invalid bytes in the FIFO 0b000000000000 No free bytes in the FIFO The maximum value returned is never greater than the FIFO Depth	The reset value for this field depends on the MHU configuration.

Accessibility

Table 5-176: Accessibility

Component	Offset	Range
PFFCW<n>_PAY32	[(64 * n) + 0x2000]	63:0

Component	Offset	Range
PFFCW<n>_PAY32	[(64 * n) + 0x2004]	63:0

5.4.2.32 PFFCW<n>_FLG, Postbox FIFO Channel Window <n> Flag Register, n = 0 - 63

Provides access to the flags of the entry of the FIFO pointed to by the write pointer.

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_FLG are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(64 * n) + 0x2008

Bit descriptions

Figure 5-88: MHUS.PBX_PFFCW<n>_FLG bit assignments

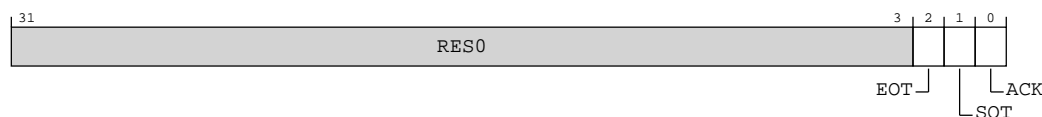


Table 5-178: PFFCW<n>_FLG bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0
[2]	EOT	<p>EOT flag.</p> <p>The EOT field indicates that the next push operation to the FIFO will contain the last byte of a Transfer.</p> <p>The behavior of this field depends on the value of the PFFCW<n>_CTRL.TDM field as follows:</p> <p>0b0</p> <p>Next push operation does not contain the last byte of the Transfer</p> <p>0b1</p> <p>Next push operation does contain the last byte of the Transfer</p> <p>For more information, see Table 5-179: PFFCW<n>_CTRL.TDM description on page 147. The EOT flag is always associated with the last byte to be pushed onto the FIFO.</p> <p>When multiple bytes are pushed onto the FIFO via a single write to the PFFCW<n>_PAY register, which byte is associated with which fields depends on the value of the PFFCW<n>_CTRL.MSBF field as follows:</p> <ul style="list-style-type: none"> 0b0 - LSB is associated with the SOT flag and MSB is associated with the EOT and ACK flags 0b1 - MSB is associated with the SOT flag and LSB is associated with the EOT and ACK flags <p>Where B0 is the lowest offset in the PFFCW<n>_PAY accesses by the write and Bn is the highest offset in the PFFCW<n>_PAY accessed by the write.</p> <p>When PFFCW<n>_CTRL.TDM == '00'</p> <p>Access to this field is: RW</p> <p>When PFFCW<n>_CTRL.TDM == '01' and (EOT field will be set to == '1' or SOT field will be set to == '1')</p> <p>Access to this field is: RW</p> <p>When PFFCW<n>_CTRL.TDM == '01' and (EOT field will be set to == '0' and SOT field will be set to == '0')</p> <p>Access to this field is: RO</p> <p>When PFFCW<n>_CTRL.TDM == '10'</p> <p>Access to this field is: RO</p>	0b0

Bits	Name	Description	Reset
[1]	SOT	<p>SOT flag.</p> <p>The SOT flag indicates that the next push operation to the FIFO will contain the first byte of a Transfer.</p> <p>The behavior of this field depends on the value of the PFFCW<n>_CTRL.TDM field as follows:</p> <p>0b0</p> <p>Next push operation does not contain the first byte of the Transfer</p> <p>0b1</p> <p>Next push operation does contain the first byte of the Transfer</p> <p>For more information, see Table 5-180: PFFCW<n>_CTRL.TDM description table 3 on page 148. The SOT flag is always associated with the first byte to be pushed onto the FIFO.</p> <p>When multiple bytes are pushed onto the FIFO via a single write to the PFFCW<n>_PAY register, which byte is associated with which fields depends on the value of the PFFCW<n>_CTRL.MSBF field as follows:</p> <ul style="list-style-type: none"> 0b0 - LSB is associated with the SOT flag and MSB is associated with the EOT and ACK flags. 0b1 - MSB is associated with the SOT flag and LSB is associated with the EOT and ACK flags. <p>Where B0 is the lowest offset in the PFFCW<n>_PAY accesses by the write and Bn is the highest offset in the PFFCW<n>_PAY accessed by the write.</p> <p>When PFFCW<n>_CTRL.TDM == '00'</p> <p>Access to this field is: RW</p> <p>When PFFCW<n>_CTRL.TDM == '01' and (EOT field will be set to == '1' or SOT field will be set to == '1')</p> <p>Access to this field is: RW</p> <p>When PFFCW<n>_CTRL.TDM == '01' and (EOT field will be set to == '0' and SOT field will be set to == '0')</p> <p>Access to this field is: RO</p> <p>When PFFCW<n>_CTRL.TDM == '10'</p> <p>Access to this field is: RO</p>	0b1
[0]	ACK	<p>ACK Flag.</p> <p>The ACK flag requests that when the Receiver pops the byte and the byte is the last byte of the Transfer, from the FIFO, a Transfer Acknowledge event is generated.</p> <p>The behavior of the ACK field is not effected by the value of PFFCW<n>_CTRL.TDM.</p> <p>0b0</p> <p>Entry is not requested to generate a Transfer Acknowledge event when popped from the FIFO</p> <p>0b1</p> <p>Entry is requested to generate a Transfer Acknowledge event when popped from the FIFO</p> <p>The ACK flag is always associated with the same byte that is associated with the EOT flag.</p>	0b0

Table 5-179: PFFCW<n>_CTRL.TDM description

PFFCW<n>_CTRL.TDM	Behavior of the EOT field
0b00	Software manages the EOT field directly and hardware never changes the value. The access permission of this field is read-write

PFFCW<n>_CTRL.TDM	Behavior of the EOT field
0b01	The EOT field is set by software and hardware. The EOT is set to 0b0 when PFFCW<n>_CTRL.TDM is set to 0b01 and can only be written to if it or the SOT field is to be set to 0b1, otherwise the value remains unchanged. When one or more bytes are pushed onto the FIFO, the EOT field is set to 0b0 irrespective of the value of the SOT or EOT fields before the push. The access permission of this field is read-write if either this field or the SOT field will be set to 0b1 by the write, otherwise it is read-only.
0b10	The EOT field is managed by hardware. The EOT flag is set to 0b0 when PFFCW<n>_CTRL.TDM is set to 0b10 and is read-only. The EOT flag value toggles when one or more bytes are pushed onto the FIFO. The access permission of this field is read-only.

Table 5-180: PFFCW<n>_CTRL.TDM description table 3

PFFCW<n>_CTRL.TDM	Behavior of the SOT field
0b00	Software manages the SOT field directly and hardware never changes the value. The access permission of this field is read-write.
0b01	The SOT field is set by software and hardware. The SOT is set to 0b1 when PFFCW<n>_CTRL.TDM is set to 0b01 and can only be written to if it or the EOT field is to be set to 0b1, otherwise the value remains unchanged. When one or more bytes are pushed onto the FIFO, the SOT field value will be set to the value of the EOT field when the push occurred. The access permission of this field is read-write if either this field or the SOT field will be set to 0b1 by the write, otherwise it is read-only.
0b10	The SOT field is managed by hardware. The SOT flag is set to 0b0 when PFFCW<n>_CTRL.TDM is set to 0b10 and is read-only. The SOT flag value toggles when one or more bytes are pushed onto the FIFO. The access permission of this field is read-only.

Accessibility

Table 5-181: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	$(64 * n) + 0x2008$	PFFCW<n>_FLG	None

5.4.2.33 PFFCW<n>_INT_ST, Postbox FIFO Channel Window <n> Interrupt Status Register, n = 0 - 63

Returns FIFO channel interrupt status

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_INT_ST are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

$(64 * n) + 0x2010$

Bit descriptions

Shows the status of the events of the DBCH

Figure 5-89: MHUS.PBX_PFFCW<n>_INT_ST bit assignments

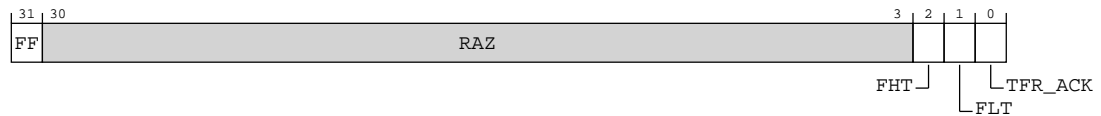


Table 5-182: PFFCW<n>_INT_ST bit descriptions

Bits	Name	Description	Reset
[31]	FF	0b0 No FIFO flush event has occurred 0b1 FIFO flush event has occurred	0b0
[30:3]	RAZ	Reserved	RAZ
[2]	FHT	0b0 No FIFO High Tidemark event has occurred 0b1 FIFO High Tidemark event has occurred	0b0
[1]	FLT	0b0 No FIFO Low Tidemark event has occurred 0b1 FIFO Low Tidemark event has occurred	0b0
[0]	TFR_ACK	0b0 No Transfer Acknowledge event 0b1 Transfer Acknowledge event	0b0

Accessibility

Table 5-183: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(64 * n) + 0x2010	PFFCW<n>_INT_ST	None

5.4.2.34 PFFCW<n>_INT_CLR, Postbox FIFO Channel Window <n> Interrupt Clear Register, n = 0 - 63

Register for clearing FIFO channel interrupt status

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_INT_CLR are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(64 * n) + 0x2014

Bit descriptions

Shows the status of the events of the DBCH

Figure 5-90: MHUS.PBX_PFFCW<n>_INT_CLR bit assignments

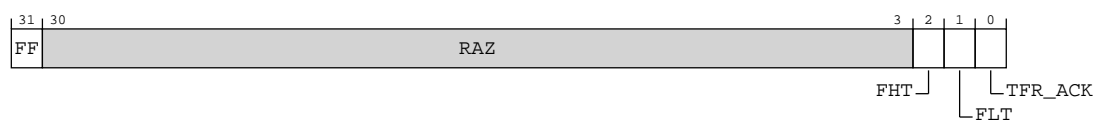


Table 5-184: PFFCW<n>_INT_CLR bit descriptions

Bits	Name	Description	Reset
[31]	FF	0b0 No effect 0b1 Clear FIFO flush event	x
[30:3]	RAZ	Reserved	RAZ
[2]	FHT	0b0 No effect 0b1 Clear FIFO High Tidemark event	0b0
[1]	FLT	0b0 No effect 0b1 Clear FIFO Low Tidemark event	0b0
[0]	TFR_ACK	0b0 No effect 0b1 Clear Transfer Acknowledge event	0b0

Accessibility

Table 5-185: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(64 * n) + 0x2014	PFFCW<n>_INT_CLR	None

5.4.2.35 PFFCW<n>_INT_EN, Postbox FIFO Channel Window <n> Interrupt Enable Register, n = 0 - 63

Register for configuring doorbell channel interrupt enables

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_INT_EN are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

$(64 * n) + 0x2018$

Bit descriptions

Shows the status of the events of the DBCH

Figure 5-91: MHUS.PBX_PFFCW<n>_INT_EN bit assignments

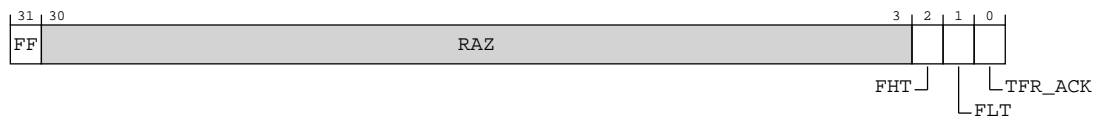


Table 5-186: PFFCW<n>_INT_EN bit descriptions

Bits	Name	Description	Reset
[31]	FF	0b0 FIFO flush event do not generate an interrupt 0b1 FIFO flush event generate an interrupt	0b1
[30:3]	RAZ	Reserved	RAZ
[2]	FHT	0b0 FIFO High Tidemark events do not generate an interrupt 0b1 FIFO High Tidemark events generate an interrupt	0b0
[1]	FLT	0b0 FIFO Low Tidemark events do not generate an interrupt 0b1 FIFO Low Tidemark events generate an interrupt	0b0

Bits	Name	Description	Reset
[0]	TFR_ACK	0b0 Transfer Acknowledge events do not generate an interrupt 0b1 Transfer Acknowledge events generate an interrupt	0b0

Accessibility

Table 5-187: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	$(64 * n) + 0x2018$	PFFCW<n>_INT_EN	None

5.4.2.36 PFFCW<n>_CTRL, Postbox FIFO Channel Window <n> Control Register, n = 0 - 63

This register contains control bits for FIFO channels

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_CTRL are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

$(64 * n) + 0x2020$

Bit descriptions

Figure 5-92: MHUS.PBX_PFFCW<n>_CTRL bit assignments

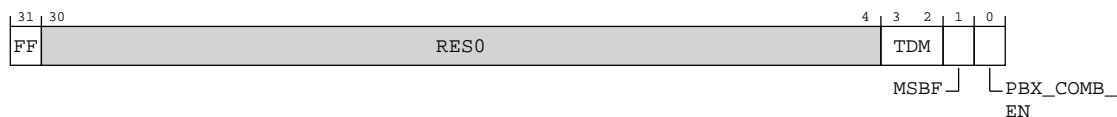


Table 5-188: PFFCW<n>_CTRL bit descriptions

Bits	Name	Description	Reset
[31]	FF	FIFO Flush Request a flush of the FFCH 0b0 No request to flush the FIFO 0b1 Request to flush the FIFO	0b0
[30:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:2]	TDM	<p>Transfer Delineation Mode</p> <p>Selects which Transfer delineation mode the MHU uses for the Channel</p> <p>Transfer delineation mode selects whether the MHU or software or a combination of both manages the SOT and EOT flags</p> <p>0b00</p> <p>None</p> <p>Software is responsible for managing the SOT and EOT flags</p> <p>0b01</p> <p>Partial</p> <p>Flags are managed partially by software and partially by the MHU</p> <p>Upon selecting Partial Transfer delineation mode the PFFCW<n>_FLG.{SOT,EOT} fields are set to 0b1 and 0b0 respectively.</p> <p>No change occurs to the PFFCW<n>_FLG.ACK field</p> <p>To update either the PFFCW<n>_FLG.{SOT/EOT} fields, at least one of them must be set to 0b1, otherwise the update to those fields are ignored</p> <p>When a push operation occurs the SOT and EOT fields are updated as follows:</p> <ul style="list-style-type: none"> • SOT field is set to value of the EOT field before the push • EOT field is always set to 0b0 <p>To update only the value of the PFFCW<n>_FLG.ACK field when using Partial Transfer delineation mode software should set both the SOT and EOT fields to 0b0 as this will keep EOT and SOT unchanged.</p> <p>0b10</p> <p>Auto</p> <p>Flags are fully managed by the MHU</p> <p>Upon selecting Auto Transfer delineation mode the PFFCW<n>_FLG.{SOT,EOT} fields are set to 0b1 and 0b0 respectively</p> <p>No change occurs to the PFFCW<n>_FLG.ACK field</p> <p>Any writes to the PFFCW<n>_FLG are ignored</p> <p>When a push operation occurs the values of the SOT and EOT fields toggle.</p> <p>To update the value of the PFFCW<n>_FLG.ACK field when using the Auto Transfer delineation mode software can write any value to the SOT and EOT fields as it will be ignored.</p> <p>All other values are Reserved</p>	0b00

Bits	Name	Description	Reset
[1]	MSBF	<p>Most Significant Byte First</p> <p>Selects the order in which bytes are pushed onto the FIFO when multiple bytes are to be pushed due to a single write to the PFFCW<n>_PAY register</p> <p>0b0</p> <p>Least Significant Byte first</p> <p>0b1</p> <p>Most Significant Byte first</p> <p>FFCH are considered little-endian and the LSB is the lowest byte offset and the MSB is the highest byte offset within the access</p>	0b0
[0]	PBX_COMB_EN	<p>0b0</p> <p>FIFO Channel does not contribute to the Postbox Combined interrupt</p> <p>0b1</p> <p>FIFO Channel contributes to the Postbox Combined interrupt</p>	0b1

Accessibility

Table 5-189: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(64 * n) + 0x2020	PFFCW<n>_CTRL	None

5.4.2.37 PFFCW<n>_ST, Postbox FIFO Channel Window <n> Status Register, n = 0 - 63

Contains status information for FIFO channel

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_ST are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(64 * n) + 0x2024

Bit descriptions

Figure 5-93: MHUS.PBX_PFFCW<n>_ST bit assignments

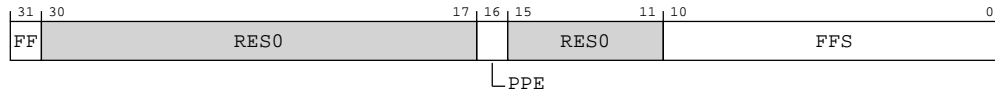


Table 5-190: PFFCW<n>_ST bit descriptions

Bits	Name	Description	Reset
[31]	FF	<p>Status of Sender FIFO flush mechanism.</p> <p>0b0</p> <p>The meaning of this value depends on the value of the FF field in the corresponding PFFCW<n>_CTRL register.</p> <ul style="list-style-type: none"> When PFFCW_CTRL<n>.FF is 0b0, the Sender FIFO flush mechanism is idle. When PFFCW_CTRL<n>.FF is 0b1, the Sender FIFO flush mechanism is performing a flush. <p>0b1</p> <p>Sender FIFO flush completed.</p>	0b0
[30:17]	RES0	Reserved	RES0
[16]	PPE	<p>Previous Push Error.</p> <p>Indicates whether a previous push to the FIFO caused an error.</p> <p>An error is when the previous push operation did not push all the bytes requested onto the FIFO.</p> <p>The reason for why bytes could not be pushed onto the FIFO are:</p> <ul style="list-style-type: none"> Not enough space in the FIFO The write to the PFFCW_PAY register did not meet the access conditions for the PFFCW<n>_PAY register for this implementation of the MHU. <p>0b0</p> <p>No error has occurred on the last push operation.</p> <p>0b1</p> <p>An error has occurred on the last push operation.</p> <p>If there has been no previous push operation to the FIFO this field is 0b0.</p>	0b0
[15:11]	RES0	Reserved	RES0
[10:0]	FFS	<p>FIFO Free Space.</p> <p>Indicates the number of invalid bytes in the FIFO. The maximum value returned is never greater than the FIFO Depth.</p>	11 {x}

Accessibility

Table 5-191: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(64 * n) + 0x2024	PFFCW<n>_ST	None

5.4.2.38 PFFCW<n>_ACK_CNT, Postbox FIFO Channel Window <n> Acknowledge Counter Register, n = 0 - 63

Allows determining the number of acknowledged FIFO channel transfers

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_ACK_CNT are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

(64 * n) + 0x2028

Bit descriptions

Figure 5-94: MHUS.PBX_PFFCW<n>_ACK_CNT bit assignments



Table 5-192: PFFCW<n>_ACK_CNT bit descriptions

Bits	Name	Description	Reset
[31:12]	RES0	Reserved	RES0
[11]	ACK_CNT_OVRFLW	Indicate whether the Acknowledge counter has overflown. 0b0 Acknowledge counter has not overflown 0b1 Acknowledge counter has overflown	0b0

Bits	Name	Description	Reset
[10:0]	ACK_CNT	<p>Count of the number of Transfer Acknowledge events which have occurred since the last time the register was read.</p> <p>When value of ACK_CNT_OVRFLW is set to 1 the value in this field no longer provides an accurate count of the number of Transfer Acknowledge events.</p> <p>The maximum value of this field is calculated by $2^{\text{clog2}(\text{FFCH_DEPTH}/\text{min_transfer_size})+1} - 1$.</p> <p>Where min_transfer_size is minimum size of a Transfer in bytes which the Sender can send.</p> <p>The value of min_transfer_size is determined from the values of the P{8/16/32/64}BA_SPT fields in the PBX_FFCH_CFG0 register. Table 5-193: P8BA_SPT description on page 158</p> <p>When the counter reaches maximum value and a new Acknowledge occurs the ACK_CNT_OVRFLW is set to 0b1 and the ACK_CNT field wraps around</p>	0b0000000000

Table 5-193: P8BA_SPT description

P8BA_SPT	P16BA_SPT	P32BA_SPT	P64BA_SPT	Minimum Transfer Size (Bytes)
1	X	X	X	1
0	1	X	X	2
0	0	1	X	4
0	0	0	1	8

Accessibility

Table 5-194: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	(64 * n) + 0x2028	PFFCW<n>_ACK_CNT	None

5.4.2.39 PFFCW<n>_TIDE, Postbox FIFO Channel Window <n> Tidemark Register, n = 0 - 63

Allows configuration of the low and high tidemark thresholds for the Sender tidemark events

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to PFFCW<n>_TIDE are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offset

$(64 * n) + 0x202C$

Bit descriptions

Figure 5-95: EXT_PFFCW<n>_TIDE bit assignments

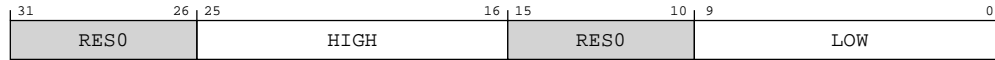


Table 5-195: PFFCW<n>_TIDE bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25:16]	HIGH	<p>High Tide Mark.</p> <p>Threshold value used in the generation of the Sender FIFO High Tide event.</p> <p>The event is generated when a push to the FIFO occurs, and the following are both true:</p> <ul style="list-style-type: none"> The FIFO fill level before the push was less than or equal to the value of this field. The FIFO fill level after the push is greater than the value of this field. <p>The upper and lower offset of this field depend on the configuration of the MHU.</p> <p>The upper offset of this field is equal to $\text{clog2}(\text{FFCH_DEPTH})+15$</p> <p>The lower offset of this field depends on the supported access sizes to PFFCW<n>_PAY register as follows:</p> <ul style="list-style-type: none"> When PBX_FFCH_CFG.P8BA_SPT is 0b1, the lower offset is 16. When PBX_FFCH_CFG.P8BA_SPT is 0b0 and PBX_FFCH_CFG.P16BA_SPT is 0b1, the lower offset is 17. When PBX_FFCH_CFG.P{8/16}BA_SPT are both 0b0 and PBX_FFCH_CFG.P32BA_SPT is 0b1, the lower offset is 18. When PBX_FFCH_CFG.P{8/16/32}BA_SPT are all 0b0 and PBX_FFCH_CFG.P64BA_SPT is 0b1, the lower offset is 19. <p>Any offsets above the upper offset or below the lower offset are RES0.</p> <p>If the upper offset is less than the lower offset, the entire field is RES0.</p> <p>In all cases the value of this field includes the RES0 offsets for calculation in the Sender High Tide event.</p>	See field description.
[15:10]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[9:0]	LOW	<p>Low Tide Mark</p> <p>Threshold value used in the generation of the Sender FIFO Low Tide event</p> <p>The event is generated when a pop to the FIFO occurs, and the following are both true:</p> <ul style="list-style-type: none"> The FIFO fill level before the pop was greater than the value of this field The FIFO fill level after the pop is less than or equal the value of this field <p>The upper and lower offset of this field depend on the configuration of the MHU.</p> <p>The upper offset of this field is equal to $\text{clog2}(\text{FFCH_DEPTH}) - 1$</p> <p>The lower offset of this field depends on the supported access sizes to PFCW<n>_PAY register as follows:</p> <ul style="list-style-type: none"> When PBX_FFCH_CFG.P8BA_SPT is 0b1, the lower offset is 0 When PBX_FFCH_CFG.P8BA_SPT is 0b0 and PBX_FFCH_CFG.P16BA_SPT is 0b1, the lower offset is 1 When PBX_FFCH_CFG.P{8/16}BA_SPT are both 0b0 and PBX_FFCH_CFG.P32BA_SPT is 0b1, the lower offset is 2 When PBX_FFCH_CFG.P{8/16/32}BA_SPT are all 0b0 and PBX_FFCH_CFG.P64BA_SPT is 0b1, the lower offset is 3 <p>Any offsets above the upper offset or below the lower offset are RES0.</p> <p>If the upper offset is less than the lower offset, the entire field is RES0.</p> <p>In all cases the value of this field includes the RES0 offsets for calculation in the Sender Low Tide event.</p>	0b0000000000

5.4.2.40 PFCW<n>_PAY32, Postbox Fast Channel Window <n> Payload 32bit Register, n = 0 - 1023

Access to payload of Fast Channel <n>

Arm recommends that accesses to these registers are atomic.
This is the 32bit version of the PFCW<n>_PAY registers

Configurations

This register is present only when FCE is implemented and FCH_WS == 0x20. Otherwise, direct accesses to PFCW<n>_PAY32 are RAZ/WI.

Attributes

Width

32

Component

MHUS.PBX

Register offsets

$(4 * n) + 0x3000$

Bit descriptions

Figure 5-96: PFCW<n>_PAY32_PFCW<n>_PAY32 bit assignments

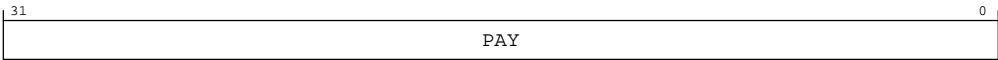


Table 5-196: PFCW<n>_PAY32 bit descriptions

Bits	Name	Description	Reset
[31:0]	PAY	Payload for Channel <n> A write to this register sets the value of the payload and generates a Transfer event A read to this register returns the current value of the payload	32 {x}

Accessibility

Table 5-197: Accessibility

Component	Offset	Range
PFCW<n>_PAY32	$[(4 * n) + 0x3000]$	1023:0

5.4.2.41 PFCW<n>_PAY64, Postbox Fast Channel Window <n> Payload 64bit Register, n = 0 - 511

Access to payload of Fast Channel <n>

Arm recommends that accesses to these registers are atomic.
This is the 64bit version of the PFCW<n>_PAY registers

Configurations

This register is present only when FCE is implemented and FCH_WS == 0x40. Otherwise, direct accesses to PFCW<n>_PAY64 are RAZ/WI.

Attributes

Width

64

Component

MHUS.PBX

Register offsets

$(8 * n) + 0x3000$

Bit descriptions

Figure 5-97: PFCW<n>_PAY64_PFCW<n>_PAY64 bit assignments

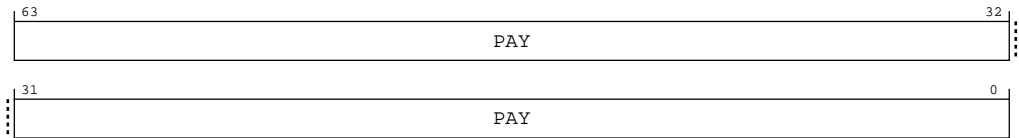


Table 5-198: PFCW<n>_PAY64 bit descriptions

Bits	Name	Description	Reset
[63:0]	PAY	Payload for Channel <n> A write to this register sets the value of the payload and generates a Transfer event A read to this register returns the current value of the payload	64 {x}

Accessibility

Table 5-199: Accessibility

Component	Offset	Range
PFCW<n>_PAY64	[(8 * n) + 0x3000]	511:0

5.4.2.42 PBX_FCTRL, Postbox Feature Control Register

Controls non-architectural postbox functionality

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.PBX

Register offset

0xF000

Bit descriptions

Figure 5-98: MHUS.PBX_PBX_FCTRL bit assignments

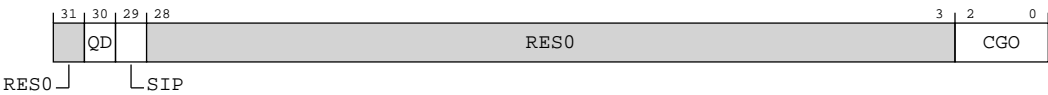


Table 5-200: PBX_FCTRL bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30]	QD	Q-Deny. Forces clock Q-channel to always deny in the Sender.	0b0
[29]	SIP	Scrub In Progress. Controls scrub of all RAMs in the Sender. 0b0 Abort scrub, if written. Scrub not in progress, if read. 0b1 Start scrub, if written. Scrub in progress, if read.	0b0
[28]	RES0	Reserved	RES0
[27]	MSIL	MHU-Stream ID Limit. Limits number of AWID values used on ACE-Lite MHU-Stream manager interface. 0b0 Use all available AWID values as per protocol. 0b1 Use only AWID=0. This field must be programmed only out of reset and not changed during operation.	0b0
[26:3]	RES0	Reserved	RES0
[2:0]	CGO	Clock Gate Override. Disables architectural clock gating. The clock gate bit assignments are: <ul style="list-style-type: none"> bit[2]: RAS register control bit[1]: FIFO channel control bit[0]: Doorbell channel control 	0b000

Accessibility

Table 5-201: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0xF000	PBX_FCTRL	None

5.4.2.43 PBX_FIFO_ERRINS, Postbox FIFO channel RAM ECC Error Insertion Register

Enables ECC error insertion in the Postbox FIFO channel RAM. The bit descriptions for this register depend on whether the access is a read or a write.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.PBX

Register offset

0xF010

Bit descriptions

When the access is a write, the PBX_FIFO_ERRINS register has the following bit assignments.

Figure 5-99: MHUS.PBX_PBX_FIFO_ERRINS write bit assignments

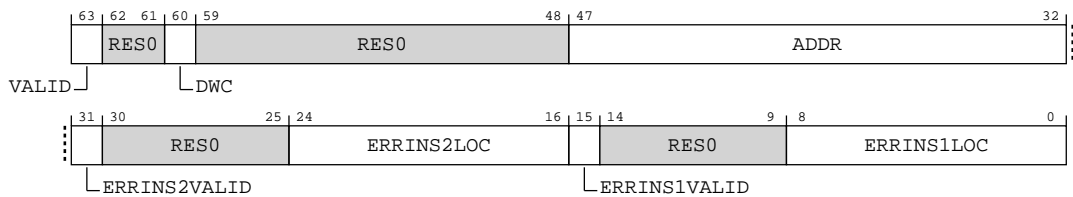


Table 5-202: PBX_FIFO_ERRINS write bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Writing 1'b1 triggers starting error insertion operation.	x
[62:61]	RES0	Reserved	RES0
[60]	DWC	Disable Write Check. If set, tests the ECC encoder.	x
[59:48]	RES0	Reserved	RES0
[47:32]	ADDR	RAM address that required error will be inserted at.	16 { x }
[31]	ERRINS2VALID	If set, enables insertion of second RAM error in bit location specified by ERRINS2LOC	x
[30:25]	RES0	Reserved	RES0
[24:16]	ERRINS2LOC	Defines bit location for second inserted RAM error	9 { x }
[15]	ERRINS1VALID	If set, enables insertion of first RAM error in bit location specified by ERRINS1LOC	x
[14:9]	RES0	Reserved	RES0
[8:0]	ERRINS1LOC	Defines bit location for first inserted RAM error	9 { x }

When the access is a read, the PBX_FIFO_ERRINS register has the following bit assignments.

Figure 5-100: MHUS.PBX_PBX_FIFO_ERRINS read bit assignments

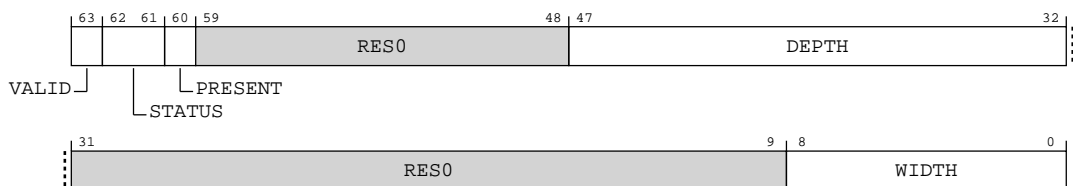


Table 5-203: PBX_FIFO_ERRINS read bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Indicates if error insertion operation is ongoing 0b0 No error insertion operation is in progress. 0b1 Error insertion operation is in progress.	0b0
[62:61]	STATUS	Result of error insertion when VALID is read as 1'b0 0b00 Error insertion was successful. 0b01 Error insertion encountered out of range address or bit location. 0b10 Error insertion overlap with real ECC error. 0b11 Encoder/decoder mismatch during error insertion.	xx
[60]	PRESENT	If set, indicates that RAM is present and allows inserting errors	x
[59:48]	RES0	Reserved	RES0
[47:32]	DEPTH	Defines the maximum RAM address for insertion	16 {x}
[31:9]	RES0	Reserved	RES0
[8:0]	WIDTH	Defines the maximum RAM bit location for insertion	9 {x}

Accessibility

Table 5-204: Accessibility

Component	Offset	Instance	Range
MHUS.PBX	0xF010	PBX_FIFO_ERRINS	None

5.4.3 MHU Sender RAS register summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MHU Sender RAS (SRAS) registers.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, see the individual field resets documented on the register description pages or the [Message Handling Unit Architecture version 3.0](#).

Table 5-205: MHUS register summary

Offset	Name	Type	Reset	Width	Description
(64 * n) + 0x0000	SRAS_ERR<n>FR	RO	See individual bit resets.	64-bit	Error Record <n> Feature Register
(64 * n) + 0x0008	SRAS_ERR<n>CTLR	RW	See individual bit resets.	64-bit	Error Record <n> Control Register
(64 * n) + 0x0010	SRAS_ERR<n>STATUS	RW	See individual bit resets.	64-bit	Error Record <n> Status Register

Offset	Name	Type	Reset	Width	Description
(64 * n) + 0x0018	SRAS_ERR<n>ADDR	RW	See individual bit resets.	64-bit	Error Record <n> Address Register
(64 * n) + 0x0020	SRAS_ERR<n>MISCO	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 0
(64 * n) + 0x0028	SRAS_ERR<n>MISC1	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 1
(64 * n) + 0x0030	SRAS_ERR<n>MISC2	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 2
(64 * n) + 0x0038	SRAS_ERR<n>MISC3	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 3
0x0E00	SRAS_ERRGSR	RO	See individual bit resets.	64-bit	Error Group Status Register
0x0E10	SRAS_ERRIIDR	RO	See individual bit resets.	32-bit	Implementation Identification Register
0x0E40	SRAS_ERRACR	RW	See individual bit resets.	64-bit	Access Configuration Register
0x0FBC	SRAS_ERRDEVARCH	RO	See individual bit resets.	32-bit	Device Architecture Register
0x0FC8	SRAS_ERRDEVID	RO	See individual bit resets.	32-bit	Device Configuration Register
0x0FD0	SRAS_ERRPIDR4	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 4 Register
0x0FD4	SRAS_ERRPIDR5	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 5 Register
0x0FD8	SRAS_ERRPIDR6	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 6 Register
0x0FDC	SRAS_ERRPIDR7	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 7 Register
0x0FE0	SRAS_ERRPIDR0	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 0 Register
0x0FE4	SRAS_ERRPIDR1	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 1 Register
0x0FE8	SRAS_ERRPIDR2	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 2 Register
0x0FEC	SRAS_ERRPIDR3	RO	See individual bit resets.	32-bit	Sender RAS Peripheral ID 3 Register
0x0FF0	SRAS_ERRCIDR0	RO	See individual bit resets.	32-bit	Sender RAS Component ID 0 Register
0x0FF4	SRAS_ERRCIDR1	RO	See individual bit resets.	32-bit	Sender RAS Component ID 1 Register
0x0FF8	SRAS_ERRCIDR2	RO	See individual bit resets.	32-bit	Sender RAS Component ID 2 Register
0x0FFC	SRAS_ERRCIDR3	RO	See individual bit resets.	32-bit	Sender RAS Component ID 3 Register

5.4.3.1 SRAS_ERR<n>FR, Error Record <n> Feature Register, n = 0 - 3

Defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

(64 * n) + 0x0000

Bit descriptions

Figure 5-101: MHUS_SRAS_ERR<n>FR bit assignments

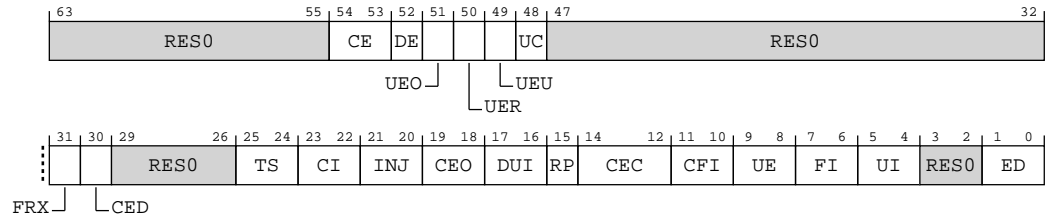


Table 5-206: SRAS_ERR<n>FR bit descriptions

Bits	Name	Description	Reset
[63:55]	RES0	Reserved	RES0
[54:53]	CE	Corrected Error recording. Describes the types of Corrected errors the node can record, if any. 0b00 Does not record Corrected errors. 0b10 Records only non-specific Corrected errors. That is, Corrected errors recorded by setting SRAS_ERR<n>STATUS.CE to 0b10.	xx
[52]	DE	Deferred Error recording. Describes whether the node supports recording Deferred errors. 0b0 Does not record Deferred errors.	0b0
[51]	UEO	Latent or Restartable Error recording. Describes whether the node supports recording Latent or Restartable errors. 0b0 Does not record Latent or Restartable errors. 0b1 Records Latent or Restartable errors.	x
[50]	UER	Signaled or Recoverable Error recording. Describes whether the node supports recording Signaled or Recoverable errors. 0b0 Does not record Signaled or Recoverable errors. 0b1 Records Signaled or Recoverable errors.	x
[49]	UEU	Unrecoverable Error recording. Describes whether the node supports recording Unrecoverable errors. 0b0 Does not record Unrecoverable errors.	0b0
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. 0b0 Does not record Uncontainable errors.	0b0
[47:32]	RES0	Reserved	RES0
[31]	FRX	Feature Register extension. Defines whether SRAS_ERR<n>FR[63:48] are architecturally defined. 0b1 SRAS_ERR<n>FR[63:48] are defined by the architecture.	0b1

Bits	Name	Description	Reset
[30]	CED	Error Counter Disable. Indicates whether the node implements a control to disable any implemented Corrected error counters. 0b0 Enabling and disabling of error counter(s) is not supported. 0b1 Enabling and disabling of error counter(s) is supported and controlled by SRAS_ERR<n>CTLR.CED.	x
[29:26]	RES0	Reserved	RES0
[25:24]	TS	Timestamp Extension. 0b00 Does not support a timestamp register.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node. 0b00 Does not support the critical error interrupt. SRAS_ERR<n>CTLR.CI is RES0.	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node. 0b00 Does not support the Common Fault Injection Model Extension.	0b00
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by this error record. 0b00 Keeps the previous error syndrome.	0b00
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node. 0b00 Does not support the enabling and disabling of error recovery interrupts on deferred errors. SRAS_ERR<n>CTLR.DUI is RES0.	0b00
[15]	RP	Repeat counter. Indicates whether the node implements a second Corrected error counter in SRAS_ERR<n>MISCO. 0b0 Implements a single Corrected error counter in SRAS_ERR<n>MISCO	0b0
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in SRAS_ERR<n>MISCO. 0b000 Does not implement an corrected error counter. 0b010 Implements an 8-bit Corrected error counter in SRAS_ERR<n>MISCO[39:32].	xxx
[11:10]	CFI	Fault handling interrupt for corrected errors control. Indicates whether the enabling and disabling of fault handling interrupts on corrected errors is supported by the node. 0b00 Does not support the enabling and disabling of fault handling interrupts on corrected errors. SRAS_ERR<n>CTLR.CFI is RES0.	0b00

Bits	Name	Description	Reset
[9:8]	UE	In-band error response (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node. 0b00 In-band error response is not supported. 0b10 In-band error response is supported and controllable using SRAS_ERR<n>CTLR.UE.	xx
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node. 0b10 Fault handling interrupt is supported and controllable using SRAS_ERR<n>CTLR.FI.	0b10
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node. 0b00 Does not support the error handling interrupt. SRAS_ERR<n>CTLR.UI is RES0 . 0b10 Error handling interrupt is supported and controllable using SRAS_ERR<n>CTLR.UI.	xx
[3:2]	RES0	Reserved	RES0
[1:0]	ED	Error reporting and logging. Indicates error record <n> is the first record owned the node, and whether the node implements the controls for enabling and disabling error reporting and logging. 0b01 Error reporting and logging always enabled. SRAS_ERR<n>CTLR.ED is RES0 .	0b01

Accessibility

Table 5-207: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	(64 * n) + 0x0000	SRAS_ERR<n>FR	None

5.4.3.2 SRAS_ERR<n>CTLR, Error Record <n> Control Register, n = 0 - 3

The error control register contains enable bits for the node that writes to this record.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

(64 * n) + 0x0008

Bit descriptions

Figure 5-102: MHUS_SRAS_ERR<n>CTLR bit assignments

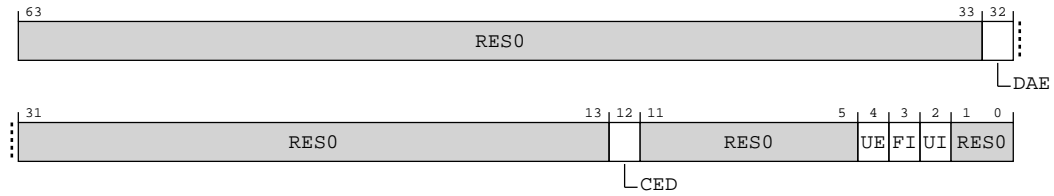


Table 5-208: SRAS_ERR<n>CTLR bit descriptions

Bits	Name	Description	Reset
[63:33]	RES0	Reserved	RES0
[32]	DAE	<p>Disable access errors. Allows disabling reporting of errors due to illegal register accesses. RAZ/WI for all records except Software error record 0.</p> <p>0b0 Illegal register accesses are treated as errors.</p> <p>0b1 Reporting of illegal register accesses is disabled.</p>	0b0
[31:13]	RES0	Reserved	RES0
[12]	CED	<p>Disable generation of corrected error events from error counters.</p> <p>0b0 Corrected error events are generated by the error counter.</p> <p>0b1 Corrected error events are generated when an error is recorded.</p>	0b0
[11:5]	RES0	Reserved	RES0
[4]	UE	<p>In-band error response enable.</p> <p>0b0 In-band error response for uncorrected errors disabled.</p> <p>0b1 In-band error response for uncorrected errors enabled.</p>	0b0
[3]	FI	<p>Fault handling interrupt enable.</p> <p>0b0 Fault handling interrupt disabled.</p> <p>0b1 Fault handling interrupt enabled.</p>	0b0
[2]	UI	<p>Uncorrected error recovery interrupt enable.</p> <p>0b0 Error recovery interrupt disabled.</p> <p>0b1 Error recovery interrupt enabled.</p>	0b0
[1:0]	RES0	Reserved	RES0

Accessibility

Table 5-209: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	$(64 * n) + 0x0008$	SRAS_ERR<n>CTLR	None

5.4.3.3 SRAS_ERR<n>STATUS, Error Record <n> Status Register, n = 0 - 3

Contains status information for error record <n>.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

$(64 * n) + 0x0010$

Bit descriptions

Figure 5-103: MHUS_SRAS_ERR<n>STATUS bit assignments

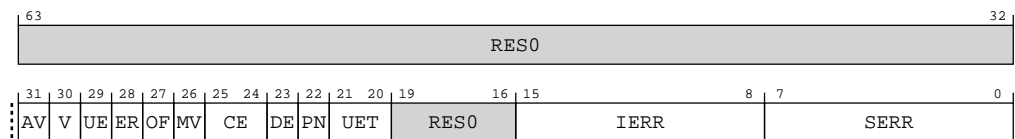


Table 5-210: SRAS_ERR<n>STATUS bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	AV	Address Valid. 0b0 SRAS_ERR<n>ADDR not valid. 0b1 SRAS_ERR<n>ADDR contains an address associated with the highest priority error recorded by this record.	0b0
[30]	V	Status Register Valid. 0b0 SRAS_ERR<n>STATUS not valid. 0b1 SRAS_ERR<n>STATUS valid. At least one error has been recorded.	0b0

Bits	Name	Description	Reset
[29]	UE	Uncorrected Error. 0b0 No errors have been detected, or all detected errors have been either corrected or deferred. 0b1 At least one detected error was not corrected and not deferred.	0b0
[28]	ER	Error Reported. 0b0 No in-band error response (External Abort) signaled to the Requester making the access. 0b1 An in-band error response was signaled by the component to the Requester making the access.	0b0
[27]	OF	Overflow. Indicates that multiple errors have been detected. 0b0 No error syndrome for an Uncorrected error has been discarded and error counter has not overflowed. 0b1 At least one error syndrome has been discarded or, if an error counter is implemented, it might have overflowed.	0b0
[26]	MV	Miscellaneous Registers Valid. 0b0 SRAS_ERR<n>MISC<m> not valid. 0b1 The SRAS_ERR<n>MISC<m> registers contain additional information for an error recorded by this record.	0b0
[25:24]	CE	Corrected Error. 0b00 No errors were corrected. 0b10 At least one error was corrected.	0b00
[23]	DE	Deferred Error. 0b0 No errors were deferred.	0b0
[22]	PN	Poison. 0b0 Error not related to a poison value.	0b0
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. 0b00 Uncorrected error, Uncontainable error (UC). 0b01 Uncorrected error, Unrecoverable error (UEU). 0b10 Uncorrected error, Latent or Restartable error (UEO). 0b11 Uncorrected error, Signaled or Recoverable error (UER).	xx
[19:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:8]	IERR	IMPLEMENTATION DEFINED error code. Used with any primary error code SRAS_ERR<n>STATUS.SERR value. See the Reliability, Accessibility and Serviceability chapter for a detailed syndrome description for each MHU error record.	0x00
[7:0]	SERR	Architecturally-defined primary error code. See the RAS System Architecture chapter of the Arm® Architecture Reference Manual for A-profile architecture for more information on this field.	0x00

Accessibility

Table 5-211: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	(64 * n) + 0x0010	SRAS_ERR<n>STATUS	None

5.4.3.4 SRAS_ERR<n>ADDR, Error Record <n> Address Register, n = 0 - 3

If an address is associated with a detected error, then it is written to ERR<n>ADDR when the error is recorded.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

(64 * n) + 0x0018

Bit descriptions

Figure 5-104: MHUS_SRAS_ERR<n>ADDR bit assignments

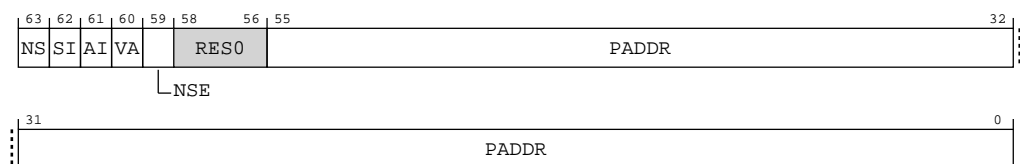


Table 5-212: SRAS_ERR<n>ADDR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. With SRAS_ERR<n>ADDR.NSE, indicates the physical address space of the recorded location. 0b0 When SRAS_ERR<n>ADDR.NSE == 0: SRAS_ERR<n>ADDR.PADDR is a Secure address. When SRAS_ERR<n>ADDR.NSE == 1: SRAS_ERR<n>ADDR.PADDR is a Root address. 0b1 When SRAS_ERR<n>ADDR.NSE == 0: SRAS_ERR<n>ADDR.PADDR is a Non-secure address. When SRAS_ERR<n>ADDR.NSE == 1: SRAS_ERR<n>ADDR.PADDR is a Realm address.	0b0
[62]	SI	Secure Incorrect. Indicates whether SRAS_ERR<n>ADDR.{NS, NSE} are valid. 0b0 SRAS_ERR<n>ADDR.{NS, NSE} are correct. That is, they match the programmers' view of the physical address space for the recorded location.	0b0
[61]	AI	Address Incorrect. Indicates whether SRAS_ERR<n>ADDR.PADDR is a valid physical address. 0b0 SRAS_ERR<n>ADDR.PADDR is a valid physical address. That is, it matches the programmers' view of the physical address for the recorded location.	0b0
[60]	VA	Virtual Address. Indicates whether SRAS_ERR<n>ADDR.PADDR field is a virtual address. 0b0 SRAS_ERR<n>ADDR.PADDR is not a virtual address.	0b0
[59]	NSE	Physical Address Space. Together with SRAS_ERR<n>ADDR.NS, indicates the address space for SRAS_ERR<n>ADDR.PADDR.	0b0
[58:56]	RES0	Reserved	RES0
[55:0]	PADDR	Physical Address. Address of the recorded location.	0x0000000000000000

Accessibility

Table 5-213: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	(64 * n) + 0x0018	SRAS_ERR<n>ADDR	None

5.4.3.5 SRAS_ERR<n>MISCO, Error Record <n> Miscellaneous Register 0, n = 0 - 3

Records information on the reported error and error counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

(64 * n) + 0x0020

Bit descriptions

Figure 5-105: MHUS_SRAS_ERR<n>MISC0 bit assignments

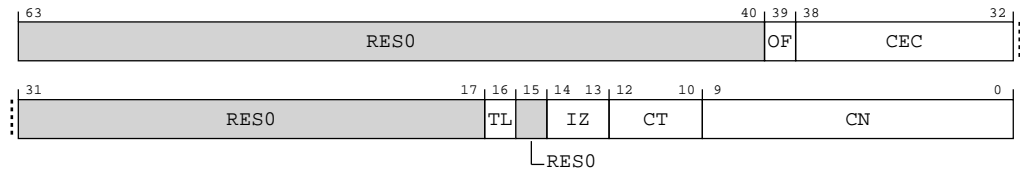


Table 5-214: SRAS_ERR<n>MISC0 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39]	OF	Sticky overflow bit. Set to 1 when SRAS_ERR<n>MISC0.CEC is incremented and wraps through zero. 0b0 Counter has not overflowed. 0b1 Counter has overflowed.	0b0
[38:32]	CEC	Error count. Incremented on for each corrected or uncorrected error.	0b0000000
[31:17]	RES0	Reserved	RES0
[16]	TL	Transfer Lost. 0b0 Error has not caused loss of transfer data in channels within the impact zone. 0b1 Error may have caused loss of transfer data in channels within the impact zone.	0b0
[15]	RES0	Reserved	RES0
[14:13]	IZ	Impact Zone. 0b00 Global. Any channel may be affected by the recorded error. 0b01 Channel specific. Only a specific channel of a given type may be affected by the recorded error as indicated by SRAS_ERR<n>MISC0.CT and SRAS_ERR<n>MISC0.CN. 0b10 Channel type. Any channel of a specific type may be affected by the recorded error as indicated by SRAS_ERR<n>MISC0.CT.	0b00

Bits	Name	Description	Reset
[12:10]	CT	Channel Type. RES0 when SRAS_ERR<n>MISC0.IZ == 2'b00 0b000 Doorbell channel. 0b001 Fast channel. 0b010 FIFO channel.	0b000
[9:0]	CN	Channel Number. RES0 when SRAS_ERR<n>MISC0.IZ != 2'b01	0b0000000000

Accessibility

Table 5-215: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	(64 * n) + 0x0020	SRAS_ERR<n>MISC0	None

5.4.3.6 SRAS_ERR<n>MISC1, Error Record <n> Miscellaneous Register 1, n = 0 - 3

Records additional information on reported error.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

(64 * n) + 0x0028

Bit descriptions

Figure 5-106: MHUS_SRAS_ERR<n>MISC1 bit assignments

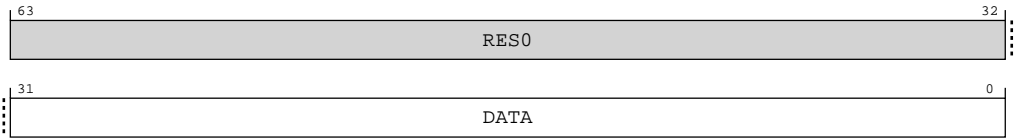


Table 5-216: SRAS_ERR<n>MISC1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:0]	DATA	Additional information on recorded error. See Reliability, Accessibility, and Serviceability , for more information.	0x00000000

Accessibility

Table 5-217: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	$(64 * n) + 0x0028$	SRAS_ERR<n>MISC1	None

5.4.3.7 SRAS_ERR<n>MISC2, Error Record <n> Miscellaneous Register 2, n = 0 - 3

Records additional information on reported error.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

$(64 * n) + 0x0030$

Bit descriptions

Figure 5-107: MHUS_SRAS_ERR<n>MISC2 bit assignments

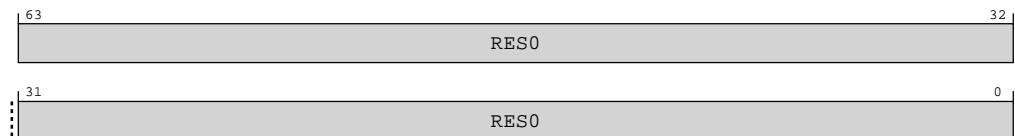


Table 5-218: SRAS_ERR<n>MISC2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	RES0

Accessibility

Table 5-219: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	$(64 * n) + 0x0030$	SRAS_ERR<n>MISC2	None

5.4.3.8 SRAS_ERR<n>MISC3, Error Record <n> Miscellaneous Register 3, n = 0 - 3

Records additional information on reported error.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

(64 * n) + 0x0038

Bit descriptions

Figure 5-108: MHUS_SRAS_ERR<n>MISC3 bit assignments

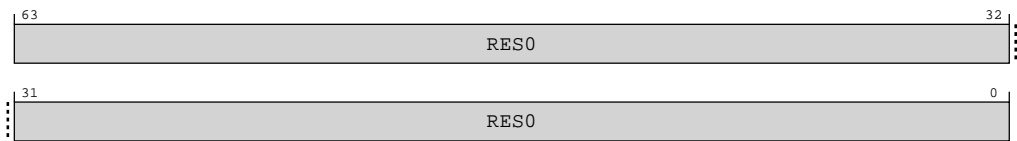


Table 5-220: SRAS_ERR<n>MISC3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	RES0

Accessibility

Table 5-221: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	(64 * n) + 0x0038	SRAS_ERR<n>MISC3	None

5.4.3.9 SRAS_ERRGSR, Error Group Status Register

Shows the status for the records in the group.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

0x0E00

Bit descriptions

Figure 5-109: MHUS_SRAS_ERRGSR bit assignments

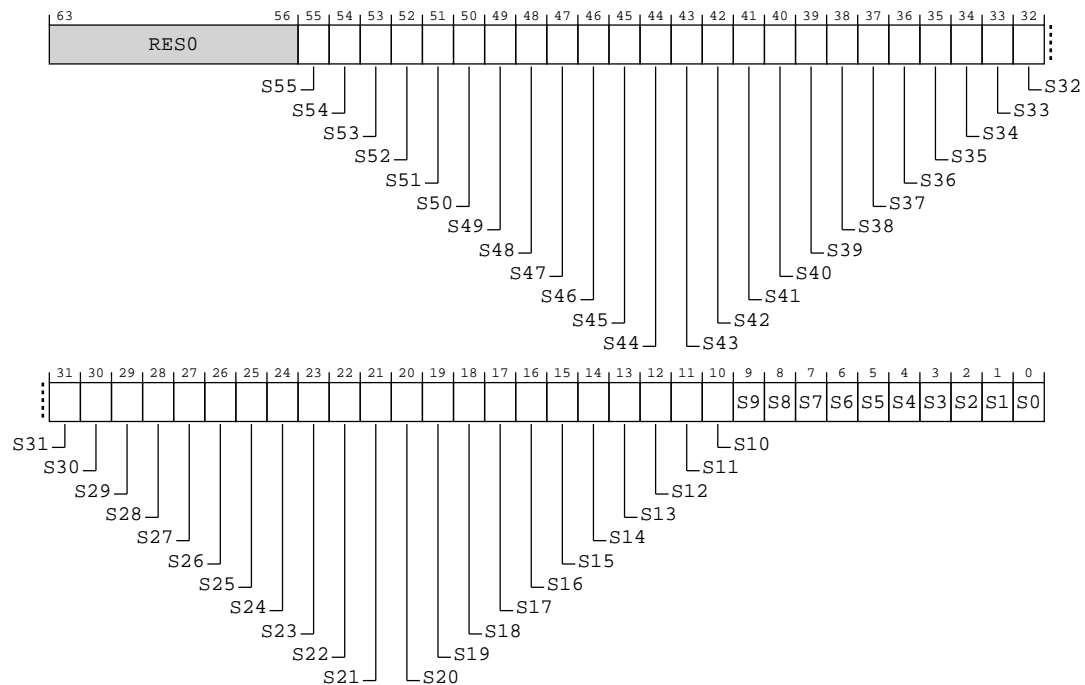


Table 5-222: SRAS_ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:0]	S<m>, bit[m], where m = 55 to 0	<p>The status for error record <m>. A read-only copy of SRAS_ERR<m>STATUS.V.</p> <p>0b0</p> <p>No error.</p> <p>0b1</p> <p>One or more errors.</p>	0x0000000000000000

Accessibility

Table 5-223: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0E00	SRAS_ERRGSR	None

5.4.3.10 SRAS_ERRIDR, Implementation Identification Register

Defines the implementer of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0x0E10

Bit descriptions

Figure 5-110: MHUS_SRAS_ERRIDR bit assignments

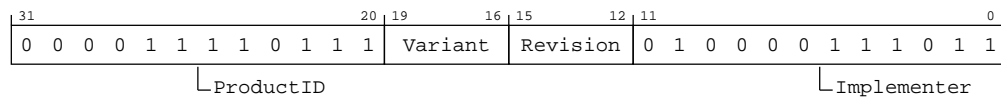


Table 5-224: SRAS_ERRIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component. 0b000011110111	0x0F7
[19:16]	Variant	Component major revision. This field distinguishes product variants or major revisions of the product.	The reset value depends on the product version used. • 0x0 - r0
[15:12]	Revision	Component minor revision. This field distinguishes minor revisions of the product.	The reset value depends on the product version used. • 0x0 - p0 • 0x1 - p1
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the RAS component. For an Arm implementation, this field has the value 0x43B. 0b010000111011	0x43B

Accessibility

Table 5-225: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0E10	SRAS_ERRIIDR	None

5.4.3.11 SRAS_ERRACR, Access Configuration Register

Controls visibility of error records.

Configurations

This register is present only when TZE is implemented for the MHU Sender. Otherwise, direct accesses to SRAS_ERRACR are RAZ/WI.

Attributes

Width

64

Component

MHUS.SRAS

Register offset

0x0E40

Bit descriptions

Figure 5-111: MHUS_SRAS_ERRACR bit assignments

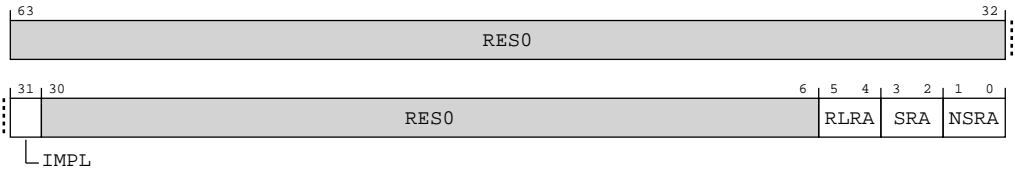


Table 5-226: SRAS_ERRACR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	IMPL	Indicates SRAS_ERRACR is present. 0b1 SRAS_ERRACR is present.	0b1
[30:6]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[5:4]	RLRA	When RME is implemented for the MHU Sender Realm Restricted Access. Controls Realm access to error records when RME is supported. 0b00 Realm access is disabled 0b01 Realm read access is enabled. Realm writes are ignored. 0b11 Realm read/write access is allowed. Otherwise RESO	xx
[3:2]	SRA	Secure Restricted Access. Controls Secure access to error records when RME is supported. 0b00 Secure access is disabled 0b01 Secure read access is enabled. Realm writes are ignored. 0b11 Secure read/write access is allowed.	0b11
[1:0]	NSRA	Non-secure Restricted Access. Controls Non-secure access to error records when TZE is supported. 0b00 Non-secure access is disabled 0b01 Non-secure read access is enabled. Realm writes are ignored. 0b11 Non-secure read/write access is allowed.	0b11

Accessibility

Table 5-227: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0E40	SRAS_ERRACR	None

5.4.3.12 SRAS_ERRDEVARCH, Device Architecture Register

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
MHUS.SRAS

Register offset
0x0FBC

Bit descriptions

Figure 5-112: MHUS_SRAS_ERRDEVARCH bit assignments

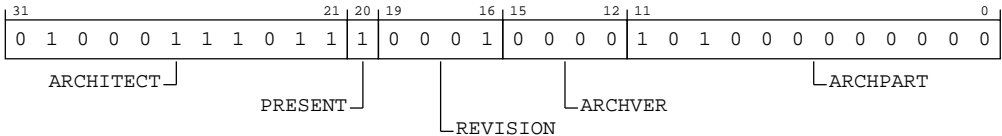


Table 5-228: SRAS_ERRDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. 0b01000111011 JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	0b01000111011
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. 0b1 Device Architecture information present.	0b1
[19:16]	REVISION	Revision. Defines the architecture revision of the component. 0b0001 RAS System Architecture v1.1.	0b0001
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. 0b0000 RAS System Architecture v1.	0b0000
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. 0b101000000000 RAS System Architecture.	0xA00

Accessibility

Table 5-229: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FBC	SRAS_ERRDEVARCH	None

5.4.3.13 SRAS_ERRDEVID, Device Configuration Register

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0x0FC8

Bit descriptions

Figure 5-113: MHUS_SRAS_ERRDEVID bit assignments

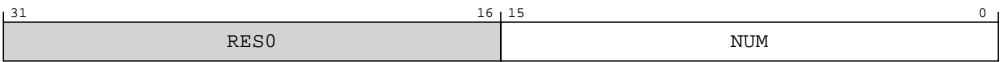


Table 5-230: SRAS_ERRDEVID bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:0]	NUM	Highest numbered index of the error records in this group, plus one.	0x4

Accessibility

Table 5-231: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FC8	SRAS_ERRDEVID	None

5.4.3.14 SRAS_ERRPIDR4, Sender RAS Peripheral ID 4 Register

Returns byte[4] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
MHUS.SRAS

Register offset
0x0FD0

Bit descriptions

Figure 5-114: MHUS_SRAS_ERRPIDR4 bit assignments

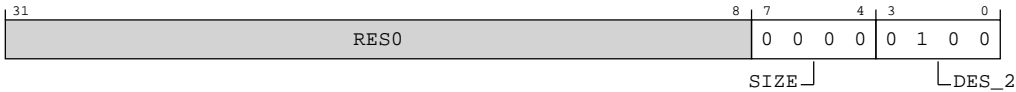


Table 5-232: SRAS_ERRPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	Size of the component 0b0000 The size of the component must be identified using a combination of the following registers: <ul style="list-style-type: none">Peripheral ID 0-7 registersComponent ID 0-3 registers<x>_FEAT_SPT0 and <x>_FEAT_SPT1 registers	0b0000
[3:0]	DES_2	JEP106 continuation code 0b0100	0b0100

Accessibility

Table 5-233: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FD0	SRAS_ERRPIDR4	None

5.4.3.15 SRAS_ERRPIDR5, Sender RAS Peripheral ID 5 Register

Returns byte[5] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUS.SRAS

Register offset
0x0FD4

Bit descriptions

Figure 5-115: MHUS_SRAS_ERRPIDR5 bit assignments

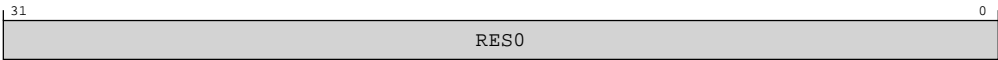


Table 5-234: SRAS_ERRPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-235: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FD4	SRAS_ERRPIDR5	None

5.4.3.16 SRAS_ERRPIDR6, Sender RAS Peripheral ID 6 Register

Returns byte[6] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUS.SRAS

Register offset
0x0FD8

Bit descriptions

Figure 5-116: MHUS_SRAS_ERRPIDR6 bit assignments

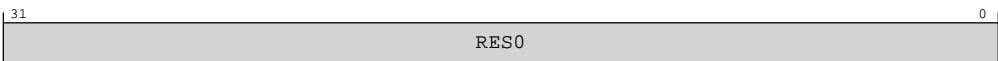


Table 5-236: SRAS_ERRPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-237: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FD8	SRAS_ERRPIDR6	None

5.4.3.17 SRAS_ERRPIDR7, Sender RAS Peripheral ID 7 Register

Returns byte[7] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0x0FDC

Bit descriptions

Figure 5-117: MHUS_SRAS_ERRPIDR7 bit assignments



Table 5-238: SRAS_ERRPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-239: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FDC	SRAS_ERRPIDR7	None

5.4.3.18 SRAS_ERRPIDR0, Sender RAS Peripheral ID 0 Register

Returns byte[0] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0x0FE0

Bit descriptions

Figure 5-118: MHUS_SRAS_ERRPIDR0 bit assignments

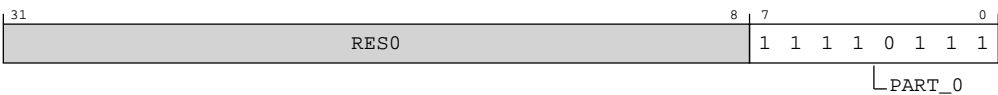


Table 5-240: SRAS_ERRPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Bits 7:0 of the Part ID for the implementation of the MHU 0b11110111	0xF7

Accessibility

Table 5-241: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FE0	SRAS_ERRPIDR0	None

5.4.3.19 SRAS_ERRPIDR1, Sender RAS Peripheral ID 1 Register

Returns byte[1] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUS.SRAS

Register offset
0x0FE4

Bit descriptions

Figure 5-119: MHUS_SRAS_ERRPIDR1 bit assignments

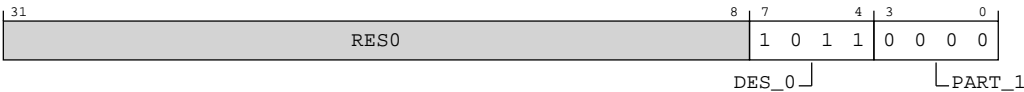


Table 5-242: SRAS_ERRPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	DES_0	Bits 3:0 of the JEP106 identification code 0b1011	0b1011
[3:0]	PART_1	Bits 11:8 of the Part ID for the implementation of the MHU 0b0000	0b0000

Accessibility

Table 5-243: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FE4	SRAS_ERRPIDR1	None

5.4.3.20 SRAS_ERRPIDR2, Sender RAS Peripheral ID 2 Register

Returns byte[2] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUS.SRAS

Register offset

0x0FE8

Bit descriptions

Figure 5-120: MHUS_SRAS_ERRPIDR2 bit assignments

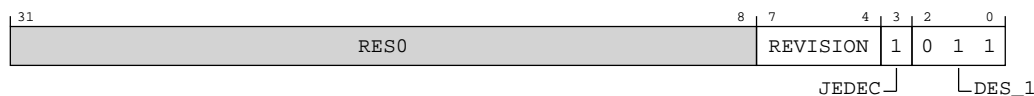


Table 5-244: SRAS_ERRPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	The REVISION field is an incremental value starting at 0x0 for the first design of a component. The value is increased by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision.	The reset value depends on the product version used. <ul style="list-style-type: none"> 0x0 - r0p0 0x1 - r0p1
[3]	JEDEC	Must be 0b1 to indicate that a JEDEC-assigned value is used. 0b1	0b1
[2:0]	DES_1	JEP106 identification bits [6:4] 0b011	0b011

Accessibility

Table 5-245: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FE8	SRAS_ERRPIDR2	None

5.4.3.21 SRAS_ERRPIDR3, Sender RAS Peripheral ID 3 Register

Returns byte[3] of the peripheral ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0x0FEC

Bit descriptions

Figure 5-121: MHUS_SRAS_ERRPIDR3 bit assignments

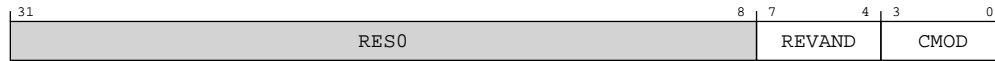


Table 5-246: SRAS_ERRPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	The REVAND field indicates minor errata fixes specific to this design, for example metal fixes after implementation. Usually this field is zero. Together with PIDR2.REVISION, PIDR3.REVAND forms the revision number of the component.	0x0
[3:0]	CMOD	Customer Modified. If the component is reusable IP, the CMOD field indicates whether the customer has modified the behavior of the component. Arm recommends that the user or debugger reads the documentation for the component to determine the modifications that are made to the component. For any two components with the same values of the Peripheral ID 0-7 and Component ID 0-3 registers <ul style="list-style-type: none"> If the value of the CMOD fields of both components equals zero, the components are identical If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have been subjected to the same modifications. If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same values of the Peripheral ID 0-7 and Component ID 0-3 registers 	0x0

Accessibility

Table 5-247: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FEC	SRAS_ERRPIDR3	None

5.4.3.22 SRAS_ERRCIDR0, Sender RAS Component ID 0 Register

Returns byte[0] of the component ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0xOFF0

Bit descriptions

Figure 5-122: MHUS_SRAS_ERRCIDR0 bit assignments

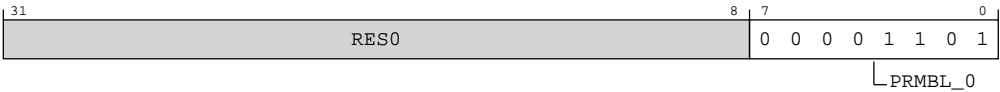


Table 5-248: SRAS_ERRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble, segment 0 0b00001101	0x0D

Accessibility

Table 5-249: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0xOFF0	SRAS_ERRCIDR0	None

5.4.3.23 SRAS_ERRCIDR1, Sender RAS Component ID 1 Register

Returns byte[1] of the component ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0xOFF4

Bit descriptions

Figure 5-123: MHUS_SRAS_ERRCIDR1 bit assignments

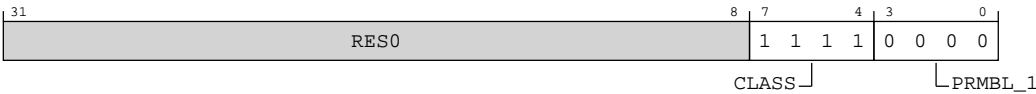


Table 5-250: SRAS_ERRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component Class 0b1111 CoreLink, PrimeCell, or system component with no standardized register layout, for backwards compatibility.	0b1111
[3:0]	PRMBL_1	Preamble, segment 1 0b0000	0b0000

Accessibility

Table 5-251: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FF4	SRAS_ERRCIDR1	None

5.4.3.24 SRAS_ERRCIDR2, Sender RAS Component ID 2 Register

Returns byte[2] of the component ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0x0FF8

Bit descriptions

Figure 5-124: MHUS_SRAS_ERRCIDR2 bit assignments

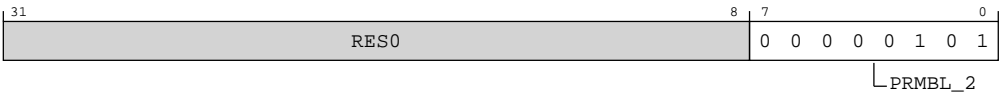


Table 5-252: SRAS_ERRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble, segment 2 0b00000101	0x05

Accessibility

Table 5-253: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0xFF8	SRAS_ERRCIDR2	None

5.4.3.25 SRAS_ERRCIDR3, Sender RAS Component ID 3 Register

Returns byte[3] of the component ID for Sender RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUS.SRAS

Register offset

0xFFC

Bit descriptions

Figure 5-125: MHUS_SRAS_ERRCIDR3 bit assignments

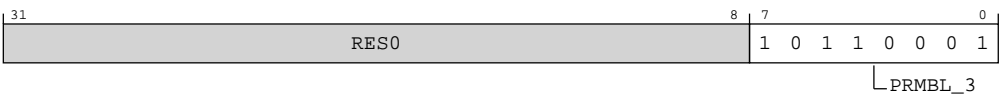


Table 5-254: SRAS_ERRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble, segment 3 0b10110001	0xB1

Accessibility

Table 5-255: Accessibility

Component	Offset	Instance	Range
MHUS.SRAS	0x0FFC	SRAS_ERRCIDR3	None

5.5 MHU Receiver registers

The MHU Receiver (MHUR) registers include the register blocks that are part of the MHU Receiver.

The offsets of the blocks, within the MHU Receiver are IMPLEMENTATION DEFINED. However, we recommend the following offsets.

- When TrustZone (TZE) is implemented for the MHUR
 - 0x0_0000 - Receiver Security Control (RSC) registers
 - 0x1_0000 - Mailbox (MBX) registers
 - 0x2_0000 - MHU Receiver RAS (RRAS) registers
- When TZE is not implemented for the MHUR
 - 0x0_0000 - MBX registers
 - 0x1_0000 - RRAS registers

5.5.1 MHU Receiver Security Control register summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MHU Receiver (MHU.RSC) registers.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, see the individual field resets documented on the register description pages or the [Message Handling Unit Architecture version 3.0](#).

Table 5-256: MHUR.RSC register summary

Offset	Name	Type	Reset	Width	Description
0x0000	RSC_BLK_ID	RO	See individual bit resets.	32-bit	Receiver Security Control Block Identifier Register

Offset	Name	Type	Reset	Width	Description
0x0010	RSC_FEAT_SPT0	RO	See individual bit resets.	32-bit	Receiver Security Feature Support 0 Register
0x0014	RSC_FEAT_SPT1	RO	See individual bit resets.	32-bit	Receiver Security Feature Support 1 Register
0x0020	RSC_DBCH_CFG0	RO	See individual bit resets.	32-bit	Receiver Security Control Doorbell Channel Configuration 0 Register
0x0030	RSC_FFCH_CFG0	RO	See individual bit resets.	32-bit	Receiver Security Control FIFO Channel Configuration 0 Register
0x0040	RSC_FCH_CFG0	RO	See individual bit resets.	32-bit	Receiver Security Control Fast Channel Configuration 0 Register
(4 * n) + 0x0110	RSC_MBX_SG<n>	RW	See individual bit resets.	32-bit	Receiver Mailbox Security Group <n> Register
0x0FC8	RSC_IIDR	RO	See individual bit resets.	32-bit	Receiver Security Implementer Identification Register
0x0FCC	RSC_AIDR	RO	See individual bit resets.	32-bit	Receiver Security Architecture Identification Register
0x0FD0	RSC_PIDR4	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 4 Register
0x0FD4	RSC_PIDR5	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 5 Register
0x0FD8	RSC_PIDR6	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 6 Register
0x0FDC	RSC_PIDR7	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 7 Register
0x0FE0	RSC_PIDR0	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 0 Register
0x0FE4	RSC_PIDR1	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 1 Register
0x0FE8	RSC_PIDR2	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 2 Register
0x0FEC	RSC_PIDR3	RO	See individual bit resets.	32-bit	Receiver Security Peripheral ID 3 Register
0x0FF0	RSC_CIDR0	RO	See individual bit resets.	32-bit	Receiver Security Component ID 0 Register
0x0FF4	RSC_CIDR1	RO	See individual bit resets.	32-bit	Receiver Security Component ID 1 Register
0x0FF8	RSC_CIDR2	RO	See individual bit resets.	32-bit	Receiver Security Component ID 2 Register
0x0FFC	RSC_CIDR3	RO	See individual bit resets.	32-bit	Receiver Security Component ID 3 Register
0xF000	RSC_ACTRL	RW	See individual bit resets.	32-bit	MHUR Access Control Register

5.5.1.1 RSC_BLK_ID, Receiver Security Control Block Identifier Register

Identifies the block as a Receiver Security Control.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_BLK_ID are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0000

Bit descriptions

Figure 5-126: MHUR.RSC_RSC_BLK_ID bit assignments

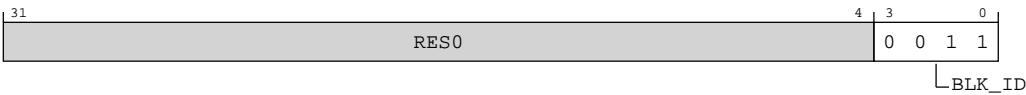


Table 5-257: RSC_BLK_ID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	BLK_ID	Block Identifier Identifies the block as a Postbox 0b0011 Receiver Security Control	0b0011

Accessibility

Table 5-258: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0000	RSC_BLK_ID	None

5.5.1.2 RSC_FEAT_SPT0, Receiver Security Feature Support 0 Register

Returns information on supported MHU features

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_FEAT_SPT0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0010

Bit descriptions

Figure 5-127: MHUR.RSC_RSC_FEAT_SPT0 bit assignments

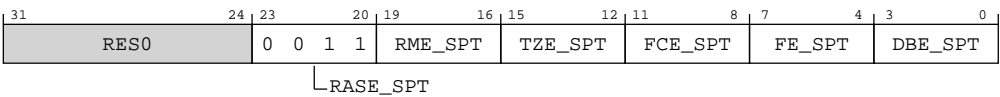


Table 5-259: RSC_FEAT_SPT0 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:20]	RASE_SPT	Reliability, Availability and Serviceability Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0011 MHU implements the RAS extension and follows the recommendations in the "Recommend implementation of RAS using Arm RAS extensions" section of Message Handling Unit Architecture version 3.0 .	0b0011

Bits	Name	Description	Reset
[19:16]	RME_SPT	<p>Realm Management Extension Support</p> <p>The value of this field depends on the implementation of the MHU and an optional reset time sampled input LEGACY_TZ_EN. The field can take one of the following values:</p> <p>0b0000 MHU does not implement the Realm Management extension</p> <p>0b0001 MHU implements Realm Management extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p> <p>When RME is implemented, for the MHU component, there can be a LEGACY_TZ_EN tie-off signal present on this MHU component.</p> <p>The value of the LEGACY_TZ_EN tie-off signal is sampled at reset of the MHU component which the tie-off signal is associated with.</p> <p>When the sampled value of the tie-off signal is 0b1 the value of this field is always 0x0, otherwise the value of this field is dependent on whether RME is implemented for this MHU component.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>
[15:12]	TZE_SPT	<p>TrustZone Extension Support</p> <p>The value of this field depends on the implementation of the MHU and can take one of the following values:</p> <p>0b0000 MHU does not implement the TrustZone extension</p> <p>0b0001 MHU implements TrustZone extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>

Bits	Name	Description	Reset
[11:8]	FCE_SPT	Fast Channel Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Fast Channel extension 0b0001 MHU implements Fast Channel extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[7:4]	FE_SPT	FIFO Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the FIFO extension 0b0001 MHU implements FIFO extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[3:0]	DBE_SPT	Doorbell Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Doorbell extension 0b0001 MHU implements Doorbell extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.

Accessibility

Table 5-260: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0010	RSC_FEAT_SPT0	None

5.5.1.3 RSC_FEAT_SPT1, Receiver Security Feature Support 1 Register

Returns information on supported MHU features

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_FEAT_SPT1 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset
0x0014

Bit descriptions

Figure 5-128: MHUR.RSC_RSC_FEAT_SPT1 bit assignments

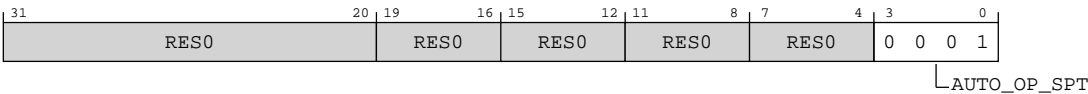


Table 5-261: RSC_FEAT_SPT1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	AUTO_OP_SPT	Auto Op Protocol support For more information about the Auto Op protocol, see the Message Handling Unit Architecture version 3.0 . The value of this field is set for MHU-320AE: 0b0001 Auto Op(Full) is implemented	0b0001

Accessibility

Table 5-262: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0014	RSC_FEAT_SPT1	None

5.5.1.4 RSC_DBCH_CFG0, Receiver Security Control Doorbell Channel Configuration 0 Register

Returns doorbell channel configuration information

Configurations

This register is present only when TZE is implemented for the MHUR and DBE is implemented. Otherwise, direct accesses to RSC_DBCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0020

Bit descriptions

Figure 5-129: MHUR.RSC_RSC_DBCH_CFG0 bit assignments

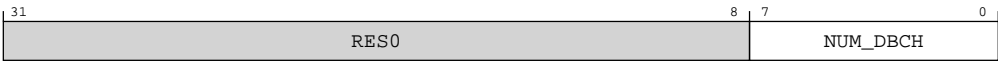


Table 5-263: RSC_DBCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	NUM_DBCH	Number of Doorbell Channels 0b00000000..0b01111111 Number of DBCH is N+1, where N is the value of this field	8 {x}

Accessibility

Table 5-264: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0020	RSC_DBCH_CFG0	None

5.5.1.5 RSC_FFCH_CFG0, Receiver Security Control FIFO Channel Configuration 0 Register

Returns FIFO channel configuration information

Configurations

This register is present only when TZE is implemented for the MHUR and FE is implemented. Otherwise, direct accesses to RSC_FFCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0030

Bit descriptions

Figure 5-130: MHUR.RSC_RSC_FFCH_CFG0 bit assignments

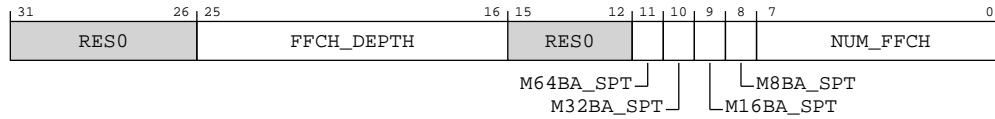


Table 5-265: RSC_FFCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25:16]	FFCH_DEPTH	FIFO Channel Depth 0b0000000000..0b1111111111 FIFO depth is N+1 bytes, where N is the value of this field	10{x}
[15:12]	RES0	Reserved	RES0
[11]	M64BA_SPT	Mailbox 64bit Access Support Whether the implementation of the MHU supports 64bit accesses to the following registers <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG 0b0 64bit accesses are not supported 0b1 64bit accesses are supported Accesses must be aligned to an 64bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.
[10]	M32BA_SPT	Mailbox 32bit Access Support Whether the implementation of the MHU supports 32bit accesses to the following registers <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG 0b0 32bit accesses are not supported 0b1 32bit accesses are supported Accesses must be aligned to an 32bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[9]	M16BA_SPT	<p>Mailbox 16bit Access Support</p> <p>Whether the implementation of the MHU supports 32bit accesses to the following registers</p> <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG <p>0b0 16bit accesses are not supported</p> <p>0b1 16bit accesses are supported</p> <p>Accesses must be aligned to an 16bit boundary</p>	The reset values can be the following: 0b0, 0b1, respective to the value.
[8]	M8BA_SPT	<p>Mailbox 8bit Access Support</p> <p>Whether the implementation of the MHU supports 8bit accesses to the following registers</p> <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG <p>0b0 8bit accesses are not supported</p> <p>0b1 8bit accesses are supported</p> <p>Accesses must be aligned to an 8bit boundary</p>	The reset values can be the following: 0b0, 0b1, respective to the value.
[7:0]	NUM_FFCH	<p>Number of FIFO Channels</p> <p>The number of FIFO Channels in the Mailbox.</p> <p>0b00000000..0b00111111 Number of FIFO is N+1, where N is the value of this field</p>	8 {x}

Accessibility

Table 5-266: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0030	RSC_FFCH_CFG0	None

5.5.1.6 RSC_FCH_CFG0, Receiver Security Control Fast Channel Configuration 0 Register

Returns fast channel configuration information

Configurations

This register is present only when TZE is implemented for the MHUR and FCE is implemented. Otherwise, direct accesses to RSC_FCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0040

Bit descriptions

Figure 5-131: MHUR.RSC_RSC_FCH_CFG0 bit assignments

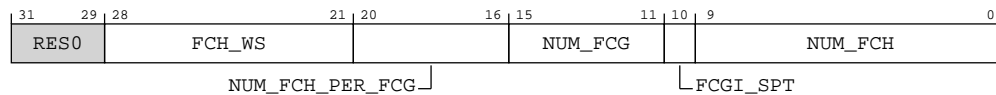


Table 5-267: RSC_FCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:29]	RES0	Reserved	RES0
[28:21]	FCH_WS	Fast Channel Word Size Number of bits each Fast Channel implements. The value must be the same as MBX_FCH_CFG0.FCH_WS. 0b00100000 Fast Channel word size is 32-bits 0b01000000 Fast Channel word size is 64-bits	The reset values can be the following: 0b00100000, 0b01000000, respective to the value.
[20:16]	NUM_FCH_PER_FCG	Number of Fast Channels per FCH Group for the Mailbox 0b000000..0b11111 Number of Fast Channels per Fast Channel Group is N+1, where N is the value of this field All other values are Reserved	The reset value for this field depends on the MHU configuration.

Bits	Name	Description	Reset
[15:11]	NUM_FCG	<p>Number of Fast Channel Groups for the Mailbox</p> <p>The number of Fast Channel Groups implemented is 1 plus the value of this field</p> <p>0b000000..0b111111</p> <p>Number of Fast Channel Groups is N+1, where N is the value of this field</p> <p>The legal values for this field are 0-31</p>	The reset value for this field depends on the MHU configuration.
[10]	FCGI_SPT	<p>Fast Channel Group Interrupt Support</p> <p>Indicates whether the MHU implementation implements the Fast Channel Group interrupt</p> <p>0b0</p> <p>Fast Channel Group Interrupts are not implemented</p> <p>0b1</p> <p>Fast Channel Group Interrupts is implemented</p> <p>The number of Fast Channel Group Interrupt is equal to the number of Fast Channel Groups, when implemented</p>	The reset values can be the following: 0b0, 0b1, respective to the value.
[9:0]	NUM_FCH	<p>Number of Fast Channels in the Mailbox</p> <p>FCH_WS == 0x40</p> <p>0b000000000000..0b011111111111</p> <p>Number of FCH is N+1, where N is the value of this field</p> <p>FCH_WS == 0x20</p> <p>0b000000000000..0b111111111111</p> <p>Number of FCH is N+1, where N is the value of this field</p>	10 {x}

Accessibility

Table 5-268: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0040	RSC_FCH_CFG0	None

5.5.1.7 RSC_MBX_SG<n>, Receiver Mailbox Security Group <n> Register, n = 0

Returns security configuration information. The bit description for this register depends on the value of the MHUR LEGACY_TZ_EN register and whether RME is implemented for the MHU

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_MBX_SG<n> are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

(4 * n) + 0x0110

Bit descriptions

When RME is implemented for the MHUR and the sampled value of MHUR LEGACY_TZ_EN is 0b0, the RSC_MBX_SG<n> register has the following bit assignments.

Figure 5-132: MHUR.RSC_RSC_MBX_SG<n> bit assignments

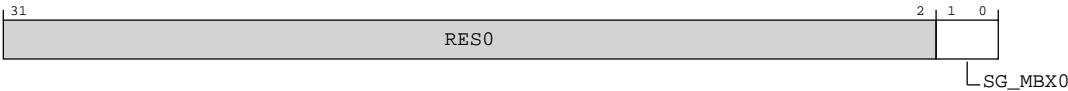


Table 5-269: RSC_MBX_SG<n> bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	SG_MBX<m>, bit[m], where m = 0	Security Group for Mailbox 0b00 Mailbox is assigned to the Secure Security Group 0b01 Mailbox is assigned to the Non-secure Security Group 0b10 Mailbox is assigned to the Root Security Group 0b11 Mailbox is assigned to the Realm Security Group	0b10

When RME is not implemented for the MHUR or RME is implemented for the MHUR and sampled value of MHUR LEGACY_TZ_EN is 0b1, the RSC_MBX_SG<n> register has the following bit assignments.

Figure 5-133: MHUR.RSC_RSC_MBX_SG<n> bit assignments

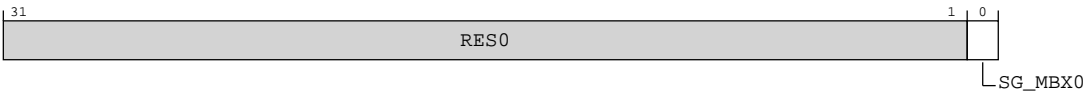


Table 5-270: RSC_MBX_SG<n> bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	SG_MBX<m>, bit[m], where m = 0	Security Group for Mailbox 0b0 Mailbox is assigned to the Secure Security Group 0b1 Mailbox is assigned to the Non-secure Security Group	0b0

Accessibility

Table 5-271: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	(4 * n) + 0x0110	RSC_MBX_SG<n>	None

5.5.1.8 RSC_IIDR, Receiver Security Implementer Identification Register

This field provides information on the Implementer of the MHU

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_IIDR are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0FC8

Bit descriptions

Figure 5-134: MHUR.RSC_RSC_IIDR bit assignments

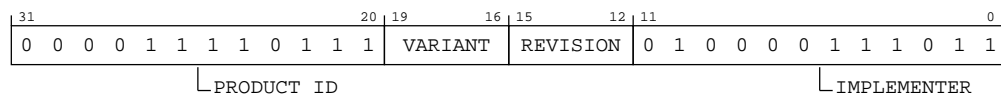


Table 5-272: RSC_IIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	PRODUCT_ID	Product ID of the MHU implementation 0b000011110111	0x0F7

Bits	Name	Description	Reset
[19:16]	VARIANT	Variant or Major revision of the MHU implementation	The reset value depends on the product version used. • 0x0 - r0
[15:12]	REVISION	Revision or minor version of the MHU implementation	The reset value depends on the product version used. • 0x0 - p0 • 0x1 - p1
[11:0]	IMPLEMENTER	Implementer ID Contains the JEP106 identification information as follows: <ul style="list-style-type: none"> 11:8 - JEP106 continuation code of implementer 7 - Always 0 6:0 - JEP106 identity code of implementer 0b010000111011	0x43B

Accessibility

Table 5-273: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FC8	RSC_IIDR	None

5.5.1.9 RSC_AIDR, Receiver Security Architecture Identification Register

Provides information on the implemented MHU architecture

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_AIDR are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0FCC

Bit descriptions

Figure 5-135: MHUR.RSC_RSC_AIDR bit assignments

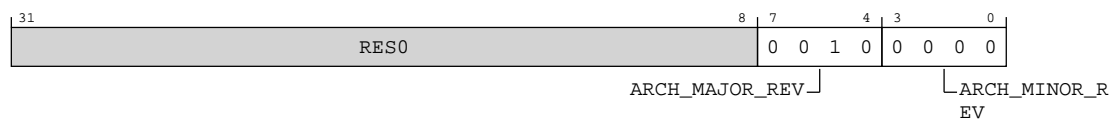


Table 5-274: RSC_AIDR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	ARCH_MAJOR_REV	MHU Architecture Major Revision 0b0010 MHUv3 All other values are Reserved	0b0010
[3:0]	ARCH_MINOR_REV	MHU Architecture Minor Revision 0b0000 Minor revision 0 of the major architecture All other values are Reserved	0b0000

Accessibility

Table 5-275: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FCC	RSC_AIDR	None

5.5.1.10 RSC_PIDR4, Receiver Security Peripheral ID 4 Register

Returns byte[4] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR4 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0FD0

Bit descriptions

Figure 5-136: MHUR.RSC_RSC_PIDR4 bit assignments

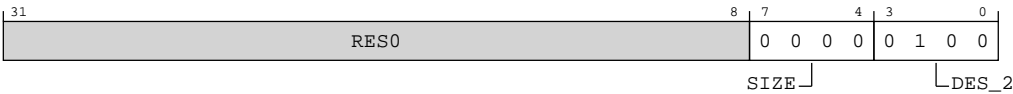


Table 5-276: RSC_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	Size of the component 0b0000 The size of the component must be identified using a combination of the following registers: <ul style="list-style-type: none">Peripheral ID 0-7 registersComponent ID 0-3 registers<x>_FEAT_SPT0 and <x>_FEAT_SPT1 registers	0b0000
[3:0]	DES_2	JEP106 continuation code 0b0100	0b0100

Accessibility

Table 5-277: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FD0	RSC_PIDR4	None

5.5.1.11 RSC_PIDR5, Receiver Security Peripheral ID 5 Register

Returns byte[5] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR5 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0FD4

Bit descriptions

Figure 5-137: MHUR.RSC_RSC_PIDR5 bit assignments

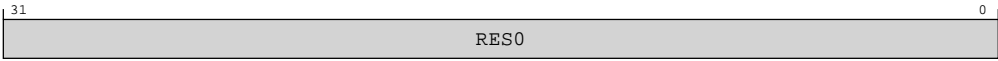


Table 5-278: RSC_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-279: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FD4	RSC_PIDR5	None

5.5.1.12 RSC_PIDR6, Receiver Security Peripheral ID 6 Register

Returns byte[6] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR6 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0FD8

Bit descriptions

Figure 5-138: MHUR.RSC_RSC_PIDR6 bit assignments

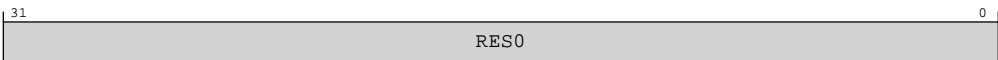


Table 5-280: RSC_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-281: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FD8	RSC_PIDR6	None

5.5.1.13 RSC_PIDR7, Receiver Security Peripheral ID 7 Register

Returns byte[7] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR7 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0FDC

Bit descriptions

Figure 5-139: MHUR.RSC_RSC_PIDR7 bit assignments

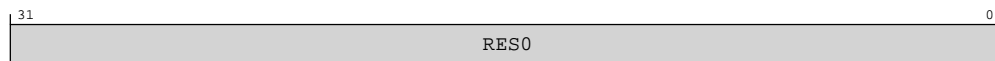


Table 5-282: RSC_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-283: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FDC	RSC_PIDR7	None

5.5.1.14 RSC_PIDR0, Receiver Security Peripheral ID 0 Register

Returns byte[0] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0x0FE0

Bit descriptions

Figure 5-140: MHUR.RSC_RSC_PIDR0 bit assignments

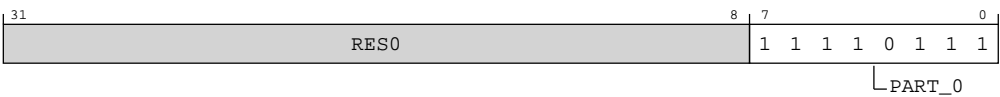


Table 5-284: RSC_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Bits 7:0 of the Part ID for the implementation of the MHU 0b11110111	0xF7

Accessibility

Table 5-285: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FE0	RSC_PIDR0	None

5.5.1.15 RSC_PIDR1, Receiver Security Peripheral ID 1 Register

Returns byte[1] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR1 are RAZ/WI.

Attributes

Width
32

Component
MHUR.RSC

Register offset
0x0FE4

Bit descriptions

Figure 5-141: MHUR.RSC_RSC_PIDR1 bit assignments



Table 5-286: RSC_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	DES_0	Bits 3:0 of the JEP106 identification code 0b1011	0b1011
[3:0]	PART_1	Bits 11:8 of the Part ID for the implementation of the MHU 0b0000	0b0000

Accessibility

Table 5-287: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FE4	RSC_PIDR1	None

5.5.1.16 RSC_PIDR2, Receiver Security Peripheral ID 2 Register

Returns byte[2] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR2 are RAZ/WI.

Attributes

Width
32

Component
MHUR.RSC

Register offset
0x0FE8

Bit descriptions

Figure 5-142: MHUR.RSC_RSC_PIDR2 bit assignments

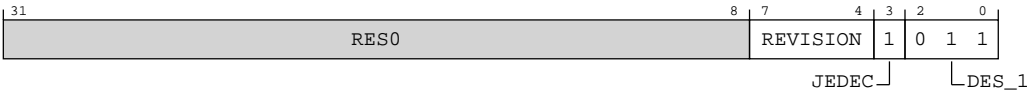


Table 5-288: RSC_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	The REVISION field is an incremental value starting at 0x0 for the first design of a component. The value is increased by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision.	The reset value depends on the product version used. <ul style="list-style-type: none">0x0 - r0p00x1 - r0p1
[3]	JEDEC	Must be 0b1 to indicate that a JEDEC-assigned value is used. 0b1	0b1
[2:0]	DES_1	JEP106 identification bits [6:4] 0b011	0b011

Accessibility

Table 5-289: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FE8	RSC_PIDR2	None

5.5.1.17 RSC_PIDR3, Receiver Security Peripheral ID 3 Register

Returns byte[3] of the peripheral ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_PIDR3 are RAZ/WI.

Attributes

Width
32

Component

MHUR.RSC

Register offset

0x0FEC

Bit descriptions

Figure 5-143: MHUR.RSC_RSC_PIDR3 bit assignments



Table 5-290: RSC_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	<p>The REVAND field indicates minor errata fixes specific to this design, for example metal fixes after implementation. Usually this field is zero.</p> <p>Together with PIDR2.REVISION, PIDR3.REVAND forms the revision number of the component.</p>	0x0
[3:0]	CMOD	<p>Customer Modified.</p> <p>If the component is reusable IP, the CMOD field indicates whether the customer has modified the behavior of the component.</p> <p>Arm recommends that the user or debugger reads the documentation for the component to determine the modifications that are made to the component.</p> <p>For any two components with the same values of the Peripheral ID 0-7 and Component ID 0-3 registers</p> <ul style="list-style-type: none"> If the value of the CMOD fields of both components equals zero, the components are identical If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have been subjected to the same modifications. If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same values of the Peripheral ID 0-7 and Component ID 0-3 registers 	0x0

Accessibility

Table 5-291: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FEC	RSC_PIDR3	None

5.5.1.18 RSC_CIDR0, Receiver Security Component ID 0 Register

Returns byte[0] of the component ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_CIDR0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0xOFF0

Bit descriptions

Figure 5-144: MHUR.RSC_RSC_CIDR0 bit assignments

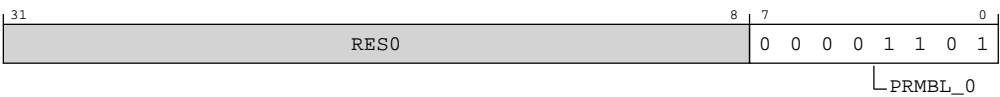


Table 5-292: RSC_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble, segment 0 0b00001101	0x0D

Accessibility

Table 5-293: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0xOFF0	RSC_CIDR0	None

5.5.1.19 RSC_CIDR1, Receiver Security Component ID 1 Register

Returns byte[1] of the component ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_CIDR1 are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0xFF4

Bit descriptions

Figure 5-145: MHUR.RSC_RSC_CIDR1 bit assignments

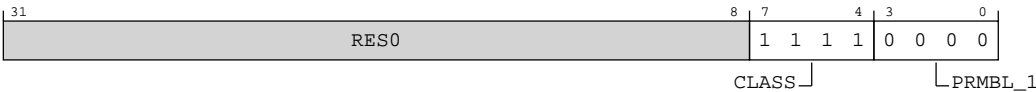


Table 5-294: RSC_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component Class 0b1111 CoreLink, PrimeCell, or system component with no standardized register layout, for backwards compatibility.	0b1111
[3:0]	PRMBL_1	Preamble, segment 1 0b0000	0b0000

Accessibility

Table 5-295: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0xFF4	RSC_CIDR1	None

5.5.1.20 RSC_CIDR2, Receiver Security Component ID 2 Register

Returns byte[2] of the component ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_CIDR2 are RAZ/WI.

Attributes

Width

32

Component
MHUR.RSC

Register offset
0x0FF8

Bit descriptions

Figure 5-146: MHUR.RSC_RSC_CIDR2 bit assignments

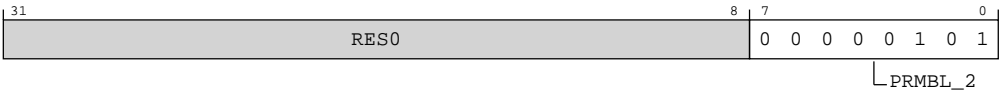


Table 5-296: RSC_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble, segment 2 0b00000101	0x05

Accessibility

Table 5-297: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FF8	RSC_CIDR2	None

5.5.1.21 RSC_CIDR3, Receiver Security Component ID 3 Register

Returns byte[3] of the component ID for Receiver Security page.

Configurations

This register is present only when TZE is implemented for the MHUR. Otherwise, direct accesses to RSC_CIDR3 are RAZ/WI.

Attributes

Width
32

Component
MHUR.RSC

Register offset
0x0FFC

Bit descriptions

Figure 5-147: MHUR.RSC_RSC_CIDR3 bit assignments

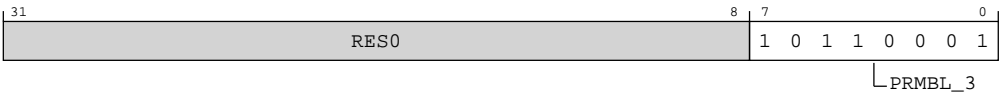


Table 5-298: RSC_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble, segment 3 0b10110001	0xB1

Accessibility

Table 5-299: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0x0FFC	RSC_CIDR3	None

5.5.1.22 RSC_ACTRL, MHUR Access Control Register

Allows overriding access control in MHUR

Configurations

This register is present only when TZE is implemented for the MHUR and RME is implemented for the MHUR. Otherwise, direct accesses to RSC_ACTRL are RAZ/WI.

Attributes

Width

32

Component

MHUR.RSC

Register offset

0xF000

Bit descriptions

Figure 5-148: MHUR.RSC_RSC_ACTRL bit assignments

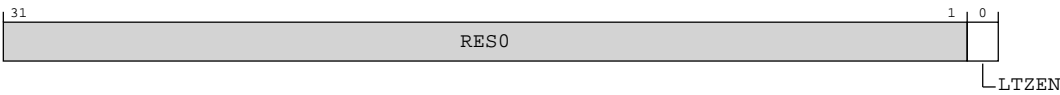


Table 5-300: RSC_ACTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	LTZEN	<p>Allows overriding LEGACY_TZ_EN tie-off</p> <p>0b0 LEGACY_TZ_EN tie-off is not set.</p> <p>0b1 LEGACY_TZ_EN tie-off is set. Reverted to TrustZone support.</p> <p>On a MHUR reset, this field resets to the LEGACY_TZ_EN tie-off value of the implementation.</p>	The reset value for this field depends on the MHU configuration.

Accessibility

Table 5-301: Accessibility

Component	Offset	Instance	Range
MHUR.RSC	0xF000	RSC_ACTRL	None

5.5.2 MHU Receiver Mailbox register summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MHU Receiver Mailbox (MHUR.MBX) registers.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, see the individual field resets documented on the register description pages or the [Message Handling Unit Architecture version 3.0](#).

Table 5-302: MHUR.MBX register summary

Offset	Name	Type	Reset	Width	Description
0x0000	MBX_BLK_ID	RO	See individual bit resets.	32-bit	Mailbox Block Identifier Register
0x0010	MBX_FEAT_SPT0	RO	See individual bit resets.	32-bit	Mailbox Feature Support 0 Register
0x0014	MBX_FEAT_SPT1	RO	See individual bit resets.	32-bit	Mailbox Feature Support 1 Register
0x0020	MBX_DBCH_CFG0	RO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Configuration 0 Register
0x0030	MBX_FFCH_CFG0	RO	See individual bit resets.	32-bit	Mailbox FIFO Channel Configuration 0 Register
0x0040	MBX_FCH_CFG0	RO	See individual bit resets.	32-bit	Mailbox Fast Channel Configuration 0 Register
0x0100	MBX_CTRL	RW	See individual bit resets.	32-bit	Mailbox Control Register

Offset	Name	Type	Reset	Width	Description
0x0140	MBX_FCH_CTRL	RW	See individual bit resets.	32-bit	Mailbox Fast Channel Control Register
0x0144	MBX_FCG_INT_EN	RW	See individual bit resets.	32-bit	Mailbox Fast Channel Group Interrupt Enable Register
(4 * n) + 0x0400	MBX_DBCH_INT_ST<n>	RO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Interrupt Status <n> Register
(4 * n) + 0x0410	MBX_FFCH_INT_ST<n>	RO	See individual bit resets.	32-bit	Mailbox FIFO Channel Interrupt Status <n> Register
0x0470	MBX_FCG_INT_ST	RO	See individual bit resets.	32-bit	Mailbox Fast Channel Group Interrupt Status Register
(4 * n) + 0x0480	MBX_FCH_GRP<n>_INT_ST	RO	See individual bit resets.	32-bit	Mailbox Fast Channel Group <n> Interrupt Status Register
0x0FC8	MBX_IIDR	RO	See individual bit resets.	32-bit	Mailbox Implementer Identification Register
0x0FCC	MBX_AIDR	RO	See individual bit resets.	32-bit	Mailbox Architecture Identification Register
0x0FD0	MBX_PIDR4	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 4 Register
0x0FD4	MBX_PIDR5	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 5 Register
0x0FD8	MBX_PIDR6	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 6 Register
0x0FDC	MBX_PIDR7	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 7 Register
0x0FE0	MBX_PIDR0	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 0 Register
0x0FE4	MBX_PIDR1	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 1 Register
0x0FE8	MBX_PIDR2	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 2 Register
0x0FEC	MBX_PIDR3	RO	See individual bit resets.	32-bit	Mailbox Peripheral ID 3 Register
0x0FF0	MBX_CIDR0	RO	See individual bit resets.	32-bit	Mailbox Component ID 0 Register
0x0FF4	MBX_CIDR1	RO	See individual bit resets.	32-bit	Mailbox Component ID 1 Register
0x0FF8	MBX_CIDR2	RO	See individual bit resets.	32-bit	Mailbox Component ID 2 Register
0x0FFC	MBX_CIDR3	RO	See individual bit resets.	32-bit	Mailbox Component ID 3 Register
(32 * n) + 0x1000	MDBCW<n>_ST	RO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Window <n> Status Register
(32 * n) + 0x1004	MDBCW<n>_ST_MSK	RO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Window <n> Status Masked Register
(32 * n) + 0x1008	MDBCW<n>_CLR	WO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Window <n> Clear Register

Offset	Name	Type	Reset	Width	Description
(32 * n) + 0x1010	MDBCW<n>_MSK_ST	RO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Window <n> Mask Status Register
(32 * n) + 0x1014	MDBCW<n>_MSK_SET	WO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Window <n> Mask Set Register
(32 * n) + 0x1018	MDBCW<n>_MSK_CLR	WO	See individual bit resets.	32-bit	Mailbox Doorbell Channel Window <n> Mask Clear Register
(32 * n) + 0x101C	MDBCW<n>_CTRL	RW	See individual bit resets.	32-bit	Mailbox Doorbell Channel Window <n> Control Register
(64 * n) + 0x2000	MFFCW<n>_PAY64	RO	See individual bit resets.	64-bit	Mailbox FIFO Channel Window <n> Payload Register (64bit access)
(64 * n) + 0x2000	MFFCW<n>_PAY32	RO	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Payload Register (32bit access)
(64 * n) + 0x2004	MFFCW<n>_PAY32	RO	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Payload Register (32bit access)
(64 * n) + 0x2008	MFFCW<n>_FLG64	RO	See individual bit resets.	64-bit	Mailbox FIFO Channel Window <n> Flag Register (64bit access)
(64 * n) + 0x2008	MFFCW<n>_FLG32	RO	See individual bit resets.	32-bit	MailboxPostbox FIFO Channel Window <n> Flag Register (32bit access)
(64 * n) + 0x200C	MFFCW<n>_FLG32	RO	See individual bit resets.	32-bit	MailboxPostbox FIFO Channel Window <n> Flag Register (32bit access)
(64 * n) + 0x2010	MFFCW<n>_INT_ST	RO	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Interrupt Status Register
(64 * n) + 0x2014	MFFCW<n>_INT_CLR	WO	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Interrupt Clear Register
(64 * n) + 0x2018	MFFCW<n>_INT_EN	RW	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Interrupt Enable Register
(64 * n) + 0x2020	MFFCW<n>_CTRL	RW	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Control Register
(64 * n) + 0x2024	MFFCW<n>_ST	RO	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Status Register
(64 * n) + 0x2028	MFFCW<n>_FIFO_POP	WO	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> FIFO POP Register
(64 * n) + 0x202C	MFFCW<n>_TIDE	RW	See individual bit resets.	32-bit	Mailbox FIFO Channel Window <n> Tidemark Register
(4 * n) + 0x3000	MFCW<n>_PAY32	RW	See individual bit resets.	32-bit	Mailbox Fast Channel Window <n> Payload 32bit Register
(8 * n) + 0x3000	MFCW<n>_PAY64	RW	See individual bit resets.	64-bit	Mailbox Fast Channel Window <n> Payload 64bit Register
0xF000	MBX_FCTRL	RW	See individual bit resets.	32-bit	Mailbox Feature Control Register
0xF010	MBX_DB_ERRINS	RW	See individual bit resets.	64-bit	Mailbox doorbell channel RAM ECC Error Insertion Register
0xF018	MBX_FST_ERRINS	RW	See individual bit resets.	64-bit	Mailbox fast channel RAM ECC Error Insertion Register
0xF020	MBX_FCFG_ERRINS	RW	See individual bit resets.	64-bit	Mailbox FIFO channel config RAM ECC Error Insertion Register

Offset	Name	Type	Reset	Width	Description
0xF028	MBX_FDATA_ERRINS	RW	See individual bit resets.	64-bit	Mailbox FIFO channel data RAM ECC Error Insertion Register

5.5.2.1 MBX_BLK_ID, Mailbox Block Identifier Register

Identifies the block as a Mailbox.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0000

Bit descriptions

Figure 5-149: MHUR.MBX_MBX_BLK_ID bit assignments

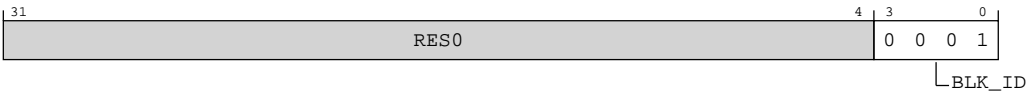


Table 5-303: MBX_BLK_ID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	BLK_ID	Block Identifier Identifies the block as a Mailbox 0b0001 Mailbox	0b0001

Accessibility

Table 5-304: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0000	MBX_BLK_ID	None

5.5.2.2 MBX_FEAT_SPT0, Mailbox Feature Support 0 Register

Returns information on supported MHU features

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0010

Bit descriptions

Figure 5-150: MHUR.MBX_MBX_FEAT_SPT0 bit assignments

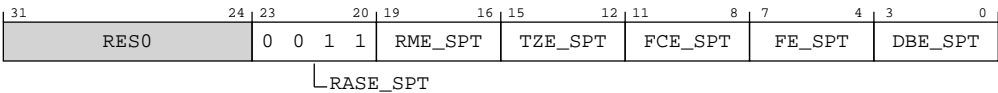


Table 5-305: MBX_FEAT_SPT0 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:20]	RASE_SPT	Reliability, Availability and Serviceability Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0011 MHU implements the RAS extension and follows the recommendations in the RAS section of the <i>Message Handling Unit Architecture version 3.0</i> .	0b0011

Bits	Name	Description	Reset
[19:16]	RME_SPT	<p>Realm Management Extension Support</p> <p>The value of this field depends on the implementation of the MHU and an optional reset time sampled input LEGACY_TZ_EN. The field can take one of the following values:</p> <p>0b0000 MHU does not implement the Realm Management extension</p> <p>0b0001 MHU implements Realm Management extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p> <p>When RME is implemented, for the MHU component, there can be a LEGACY_TZ_EN tie-off signal present on this MHU component.</p> <p>The value of the LEGACY_TZ_EN tie-off signal is sampled at reset of the MHU component which the tie-off signal is associated with.</p> <p>When the sampled value of the tie-off signal is 0b1 the value of this field is always 0x0, otherwise the value of this field is dependent on whether RME is implemented for this MHU component.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>
[15:12]	TZE_SPT	<p>TrustZone Extension Support</p> <p>The value of this field depends on the implementation of the MHU and can take one of the following values:</p> <p>0b0000 MHU does not implement the TrustZone extension</p> <p>0b0001 MHU implements TrustZone extension</p> <p>The value of this field only applies to the MHU component which the register is associated with.</p> <p>It is valid for the different MHU components to implement different values for this field.</p> <p>For fields in the PBX_FEAT_SPT0 or SSC_FEAT_SPT0 registers the value applies to the MHUS only.</p> <p>For fields in the MBX_FEAT_SPT0 or RSC_FEAT_SPT0 registers the value applies to the MHUR only.</p>	<p>The reset values can be the following: 0b0000, 0b0001, respective to the value.</p>

Bits	Name	Description	Reset
[11:8]	FCE_SPT	Fast Channel Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Fast Channel extension 0b0001 MHU implements Fast Channel extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[7:4]	FE_SPT	FIFO Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the FIFO extension 0b0001 MHU implements FIFO extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[3:0]	DBE_SPT	Doorbell Extension Support The value of this field depends on the implementation of the MHU and can take one of the following values: 0b0000 MHU does not implement the Doorbell extension 0b0001 MHU implements Doorbell extension	The reset values can be the following: 0b0000, 0b0001, respective to the value.

Accessibility

Table 5-306: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0010	MBX_FEAT_SPT0	None

5.5.2.3 MBX_FEAT_SPT1, Mailbox Feature Support 1 Register

Returns information on supported MHU features

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset
0x0014

Bit descriptions

Figure 5-151: MHUR.MBX_MBX_FEAT_SPT1 bit assignments

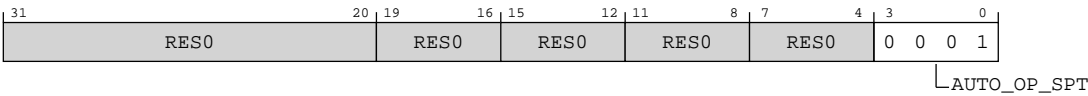


Table 5-307: MBX_FEAT_SPT1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	AUTO_OP_SPT	Auto Op Protocol support For more information about the Auto Op protocol, see the Message Handling Unit Architecture version 3.0 . The value of this field is set for MHU-320AE: 0b0001 Auto Op(Full) is implemented	0b0001

Accessibility

Table 5-308: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0014	MBX_FEAT_SPT1	None

5.5.2.4 MBX_DBCH_CFG0, Mailbox Doorbell Channel Configuration 0 Register

Returns doorbell channel configuration information

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MBX_DBCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0020

Bit descriptions

Figure 5-152: MHUR.MBX_MBX_DBCH_CFG0 bit assignments

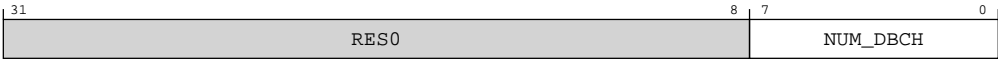


Table 5-309: MBX_DBCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	NUM_DBCH	Number of Doorbell Channels 0b00000000..0b01111111 Number of DBCH is N+1, where N is the value of this field	8{x}

Accessibility

Table 5-310: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0020	MBX_DBCH_CFG0	None

5.5.2.5 MBX_FFCH_CFG0, Mailbox FIFO Channel Configuration 0 Register

Returns FIFO channel configuration information

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MBX_FFCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0030

Bit descriptions

Figure 5-153: MHUR.MBX_MBX_FFCH_CFG0 bit assignments

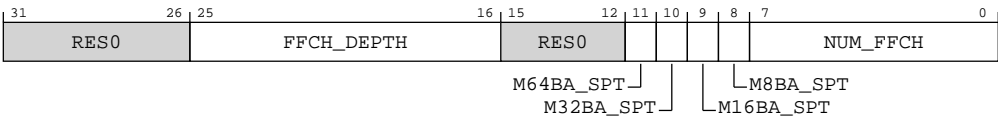


Table 5-311: MBX_FFCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25:16]	FFCH_DEPTH	FIFO Channel Depth 0b0000000000..0b1111111111 FIFO depth is N+1 bytes, where N is the value of this field	10{x}
[15:12]	RES0	Reserved	RES0
[11]	M64BA_SPT	Mailbox 64bit Access Support Whether the implementation of the MHU supports 64bit accesses to the following registers: <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG 0b0 64bit accesses are not supported 0b1 64bit accesses are supported Accesses must be aligned to an 64bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.
[10]	M32BA_SPT	Mailbox 32bit Access Support Whether the implementation of the MHU supports 32bit accesses to the following registers: <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG 0b0 32bit accesses are not supported 0b1 32bit accesses are supported Accesses must be aligned to an 32bit boundary The value of this field has no effect the supported access sizes to any other registers	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[9]	M16BA_SPT	<p>Mailbox 16bit Access Support</p> <p>Whether the implementation of the MHU supports 32bit accesses to the following registers:</p> <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG <p>0b0 16bit accesses are not supported</p> <p>0b1 16bit accesses are supported</p> <p>Accesses must be aligned to an 16bit boundary</p>	The reset values can be the following: 0b0, 0b1, respective to the value.
[8]	M8BA_SPT	<p>Mailbox 8bit Access Support</p> <p>Whether the implementation of the MHU supports 8bit accesses to the following registers:</p> <ul style="list-style-type: none"> MFFCW<n>_PAY MFFCW<n>_FLG <p>0b0 8bit accesses are not supported</p> <p>0b1 8bit accesses are supported</p> <p>Accesses must be aligned to an 8bit boundary</p>	The reset values can be the following: 0b0, 0b1, respective to the value.
[7:0]	NUM_FFCH	<p>Number of FIFO Channels</p> <p>The number of FIFO Channels in the Mailbox.</p> <p>0b00000000..0b00111111 Number of FIFO is N+1, where N is the value of this field</p>	8 {x}

Accessibility

Table 5-312: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0030	MBX_FFCH_CFG0	None

5.5.2.6 MBX_FCH_CFG0, Mailbox Fast Channel Configuration 0 Register

Returns fast channel configuration information

Configurations

This register is present only when FCE is implemented. Otherwise, direct accesses to MBX_FCH_CFG0 are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0040

Bit descriptions

Figure 5-154: MHUR.MBX_MBX_FCH_CFG0 bit assignments

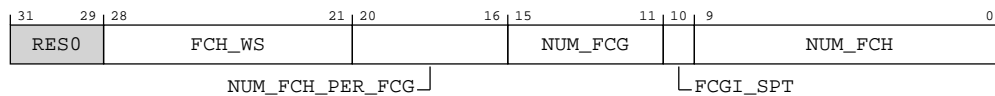


Table 5-313: MBX_FCH_CFG0 bit descriptions

Bits	Name	Description	Reset
[31:29]	RES0	Reserved	RES0
[28:21]	FCH_WS	Fast Channel Word Size Number of bits each Fast Channel implements. The value must be the same as MBX_FCH_CFG0.FCH_WS. 0b00100000 Fast Channel word size is 32-bits 0b01000000 Fast Channel word size is 64-bits	The reset values can be the following: 0b00100000, 0b01000000, respective to the value.
[20:16]	NUM_FCH_PER_FCG	Number of Fast Channels per FCH Group for the Mailbox 0b000000..0b11111 Number of Fast Channels per Fast Channel Group is N+1, where N is the value of this field All other values are Reserved	The reset value for this field depends on the MHU configuration.
[15:11]	NUM_FCG	Number of Fast Channel Groups for the Mailbox The number of Fast Channel Groups implemented is 1 plus the value of this field 0b000000..0b11111 Number of Fast Channel Groups is N+1, where N is the value of this field The legal values for this field are 0-31	The reset value for this field depends on the MHU configuration.

Bits	Name	Description	Reset
[10]	FCGI_SPT	<p>Fast Channel Group Interrupt Support</p> <p>Indicates whether the MHU implementation implements the Fast Channel Group interrupt</p> <p>0b0 Fast Channel Group Interrupts are not implemented</p> <p>0b1 Fast Channel Group Interrupts is implemented</p> <p>The number of Fast Channel Group Interrupt is equal to the number of Fast Channel Groups, when implemented</p>	The reset values can be the following: 0b0, 0b1, respective to the value.
[9:0]	NUM_FCH	<p>Number of Fast Channels in the Mailbox</p> <p>FCH_WS == 0x40 0b0000000000..0b0111111111 Number of FCH is N+1, where N is the value of this field</p> <p>FCH_WS == 0x20 0b0000000000..0b1111111111 Number of FCH is N+1, where N is the value of this field</p>	The reset value for this field depends on the MHU configuration.

Accessibility

Table 5-314: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0040	MBX_FCH_CFG0	None

5.5.2.7 MBX_CTRL, Mailbox Control Register

This register contains control bits for the mailbox

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0100

Bit descriptions

Figure 5-155: MHUR.MBX_MBX_CTRL bit assignments

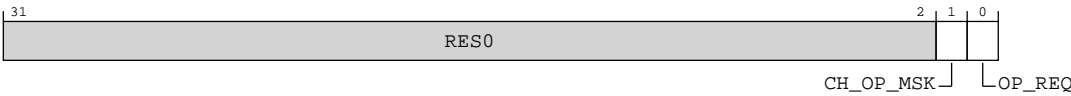


Table 5-315: MBX_CTRL bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	CH_OP_MSK	Channel Operational Mask Controls whether Channels need to be idle to allow a controlled entry into a non-operational state 0b0 Channels must be idle to enter a non-operational state 0b1 Channels status is ignored when considering entry into a non-operational state This field has no effect on entry into a non-operation state in an uncontrolled manner	0b0
[0]	OP_REQ	Operational Request 0b0 Mailbox is not requested to remain in the operational state 0b1 Mailbox is requested to remain in the operational state	0b0

Accessibility

Table 5-316: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0100	MBX_CTRL	None

5.5.2.8 MBX_FCH_CTRL, Mailbox Fast Channel Control Register

Controls the Fast Channels in the Mailbox

Configurations

This register is present only when FCE is implemented. Otherwise, direct accesses to MBX_FCH_CTRL are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset
0x0140

Bit descriptions

Figure 5-156: MHUR.MBX_MBX_FCH_CTRL bit assignments

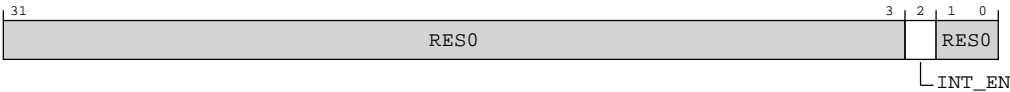


Table 5-317: MBX_FCH_CTRL bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0
[2]	INT_EN	Enables interrupts for all Fast Channels 0b0 Interrupts for Fast Channels are disabled 0b1 Interrupts for Fast Channels are enabled	0b1
[1:0]	RES0	Reserved	RES0

Accessibility

Table 5-318: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0140	MBX_FCH_CTRL	None

5.5.2.9 MBX_FCG_INT_EN, Mailbox Fast Channel Group Interrupt Enable Register

Controls whether a Fast Channel Group contributes to the Mailbox Combined interrupt

Configurations

This register is present only when FCE is implemented and FCGI_SPT. Otherwise, direct accesses to MBX_FCG_INT_EN are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0144

Bit descriptions

Figure 5-157: MHUR.MBX_MBX_FCG_INT_EN bit assignments

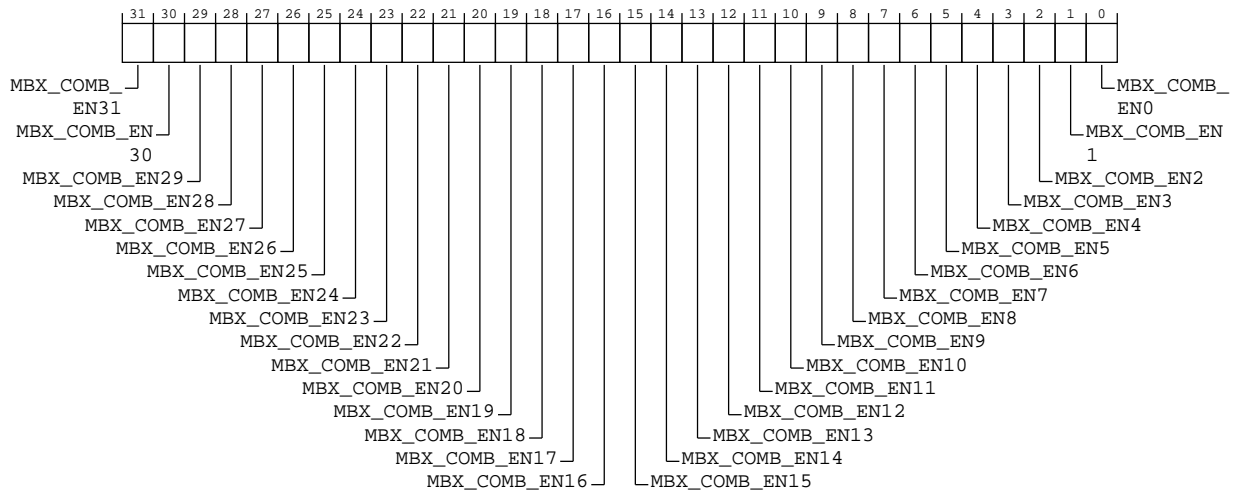


Table 5-319: MBX_FCG_INT_EN bit descriptions

Bits	Name	Description	Reset
[31:0]	MBX_COMB_EN<m>, bit[m], where m = 31 to 0	<p>0b0</p> <p>Fast Channel Group <m> does not contribute to the Mailbox Combined interrupt</p> <p>0b1</p> <p>Fast Channel Group <m> contributes to the Mailbox Combined interrupt</p> <p>The field is only implemented, if the associated Fast Channel Group is implemented. Fields which are not implemented are RES0.</p>	0xFFFFFFFF

Accessibility

Table 5-320: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0144	MBX_FCG_INT_EN	None

5.5.2.10 MBX_DBCH_INT_ST<n>, Mailbox Doorbell Channel Interrupt Status <n> Register, n = 0 - 3

Indicates whether there is an interrupt outstanding for the Doorbell Channel

MBX_DBCH_INT_ST0 has status fields for Doorbell Channels 0 to 31

MBX_DBCH_INT_ST1 has status fields for Doorbell Channels 32 to 63

MBX_DBCH_INT_ST2 has status fields for Doorbell Channels 64 to 95

MBX_DBCH_INT_ST3 has status fields for Doorbell Channels 96 to 127

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MBX_DBCH_INT_ST<n> are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(4 * n) + 0x0400$

Bit descriptions

Figure 5-158: MHUR.MBX_MBX_DBCH_INT_ST<n> bit assignments

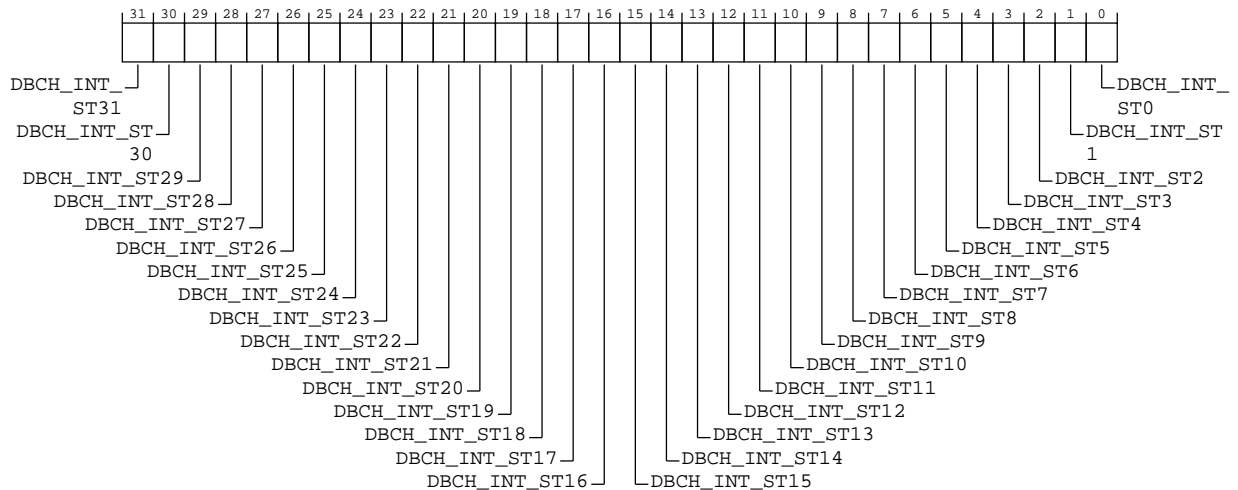


Table 5-321: MBX_DBCH_INT_ST<n> bit descriptions

Bits	Name	Description	Reset
[31:0]	DBCH_INT_ST<x>	<p>Doorbell Channel Interrupt Status</p> <p>Each bit indicates whether there is an interrupt outstanding for the Doorbell Channel</p> <p>To clear the interrupt, software must clear the underlying cause of the interrupt in the MDBCW<n>_INT_ST register using the MDBCW<n>_INT_CLR register.</p> <p>0b0</p> <p>No interrupt outstanding for the Doorbell Channel or the Channel is not configured to factor into the Mailbox Combined interrupt.</p> <p>0b1</p> <p>Interrupt outstanding for the Doorbell Channel and the Channel is configured to factor into the Mailbox Combined interrupt.</p> <p>Any fields which are not assigned to a Channel are Reserved and treated as RAZ/WI</p>	0x00000000

Accessibility

Table 5-322: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(4 * n) + 0x0400	MBX_DBCH_INT_ST<n>	None

5.5.2.11 MBX_FFCH_INT_ST<n>, Mailbox FIFO Channel Interrupt Status <n> Register, n = 0 - 1

Indicates whether there is an interrupt outstanding for the FIFO Channel.

MBX_FFCH_INT_ST0 has status fields for FIFO Channels 0 to 31

MBX_FFCH_INT_ST1 has status fields for FIFO Channels 32 to 63

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MBX_FFCH_INT_ST<n> are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(4 * n) + 0x0410

Bit descriptions

Figure 5-159: MHUR.MBX_MBX_FFCH_INT_ST<n> bit assignments

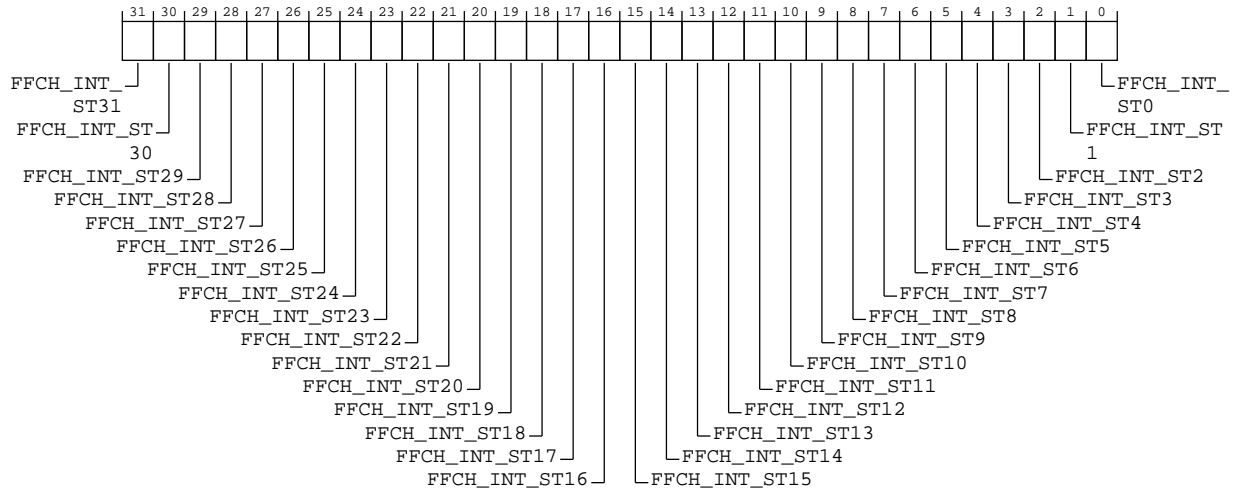


Table 5-323: MBX_FFCH_INT_ST<n> bit descriptions

Bits	Name	Description	Reset
[31:0]	FFCH_INT_ST<x>	<p>FIFO Channel Interrupt Status</p> <p>Each bit indicates whether there is an interrupt outstanding for the FIFO Channel.</p> <p>To clear the interrupt, software must clear the underlying cause of the interrupt in the MFFCW<n>_INT_ST register using the MFFCW<n>_INT_CLR register.</p> <p>0b0</p> <p>No interrupt outstanding for the FIFO Channel or the Channel is not configured to factor into the Mailbox Combined interrupt.</p> <p>0b1</p> <p>Interrupt outstanding for the FIFO Channel and the Channel is configured to factor into the Mailbox Combined interrupt.</p> <p>Any fields which are not assigned to a Channel are Reserved and treated as RAZ/WI</p>	0x00000000

Accessibility

Table 5-324: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(4 * n) + 0x0410	MBX_FFCH_INT_ST<n>	None

5.5.2.12 MBX_FCG_INT_ST, Mailbox Fast Channel Group Interrupt Status Register

Provides the status of each Fast Channel Group Transfer interrupt

Configurations

This register is present only when FCE is implemented and FCGI_SPT. Otherwise, direct accesses to MBX_FCG_INT_ST are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0470

Bit descriptions

Figure 5-160: MHUR.MBX_MBX_FCG_INT_ST bit assignments

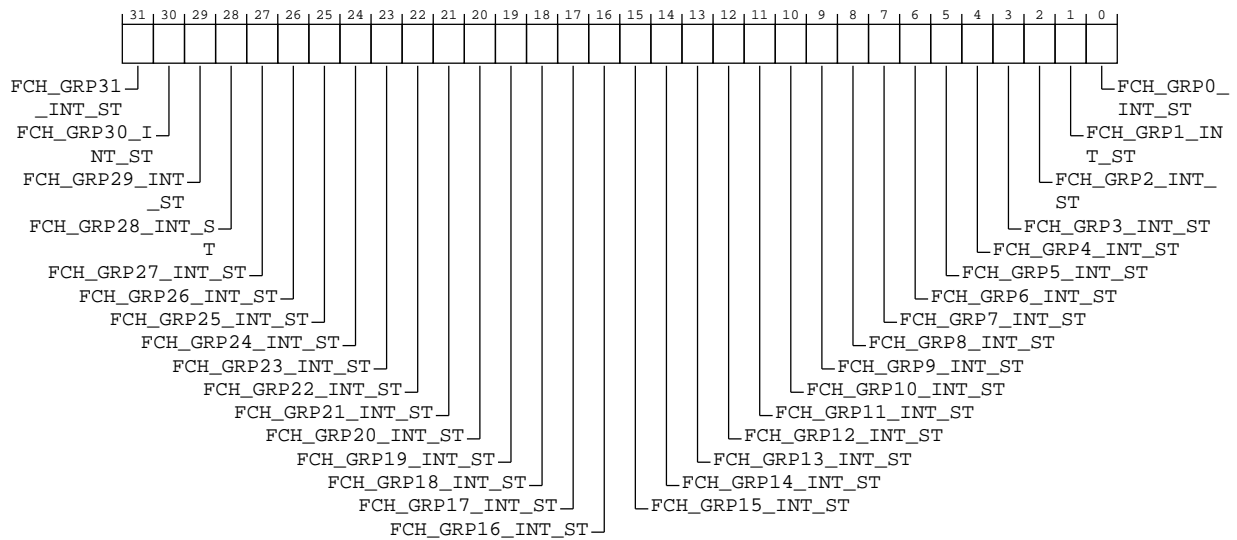


Table 5-325: MBX_FCG_INT_ST bit descriptions

Bits	Name	Description	Reset
[31:0]	FCH_GRP<m>_INT_ST, bit[m], where m = 31 to 0	<p>Indicates the status of each Fast Channel Group Transfer interrupt</p> <p>The Fast Channels which are part of each Fast Channel Group are calculated by the following formulas.</p> <p>Lowest Channel in the Group = $\text{NUM_FCH} / \text{NUM_FCH_PER_FCG} * m$</p> <p>Highest Channel in the Group = $(\text{NUM_FCH} / \text{NUM_FCH_PER_FCG} * m) + \text{NUM_FCH_PER_FCG}$</p> <p>where:</p> <ul style="list-style-type: none"> NUM_FCH - Number of Fast Channels in the Mailbox NUM_FCH_PER_FCG - Number of Fast Channels per group <p>m is the Fast Channel Group number</p> <p>To clear the interrupt, software must clear the underlying source of the interrupt in the MBX_FCH_GRP<n>_INT_ST register by acknowledging the last Transfer on the FCHs of the Fast Channel Group which indicate there is an unacknowledged Transfer.</p> <p>0b0</p> <p>No Fast Channel Group Transfer interrupt for Fast Channel Group <m> or Fast Channel Group <m> is configured not to factor into the Mailbox Combined interrupt.</p> <p>0b1</p> <p>Fast Channel Group Transfer interrupt for Fast Channel Group <m> and Fast Channel Group <m> is configured to factor into the Mailbox Combined interrupt.</p> <p>Only bits NUM_FCG-1:0 are implemented, with all unused bits being RES0</p>	0x00000000

Accessibility

Table 5-326: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0470	MBX_FCG_INT_ST	None

5.5.2.13 MBX_FCH_GRP<n>_INT_ST, Mailbox Fast Channel Group <n> Interrupt Status Register, n = 0 - 31

Provides the status of each Fast Channel with the Fast Channel Transfer Group <n>

The number of MBX_FCH_GRP<n>_INT_ST; is set by the value of MBX_FCH_CFG0.NUM_FCG, with any unused registers being **RES0**.

The number of fields within each register depends on the value of MBX_FCH_CFG0.NUM_FCH_PER_FCG, with any additional fields being **RES0**.

With bit 0 representing the status of Fast Channel m in the Mailbox and bit NUM_FCH_PER_FCG-1 representing the status of Fast Channel m+NUM_FCH_PER_FCG in the Mailbox

Where m is calculated as $n * (\text{NUM_FCH_PER_FCG})$

When all fields in the registers are **RES0** the register is also treated as **RES0**

Configurations

This register is present only when FCE is implemented and FCGI_SPT. Otherwise, direct accesses to MBX_FCH_GRP<n>_INT_ST are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(4 * n) + 0x0480$

Bit descriptions

Figure 5-161: MHUR.MBX_MBX_FCH_GRP<n>_INT_ST bit assignments

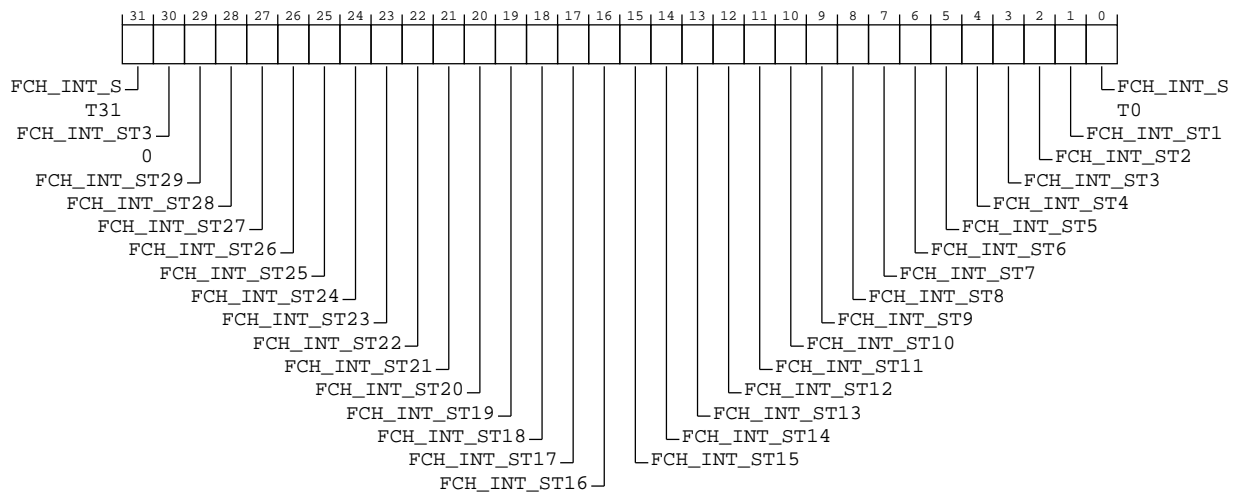


Table 5-327: MBX_FCH_GRP<n>_INT_ST bit descriptions

Bits	Name	Description	Reset
[31:0]	FCH_INT_ST<m>, bit[m], where m = 31 to 0	<p>Indicates the status of each Fast Channel within the Fast Channel Group Transfer interrupt</p> <p>The Fast Channel the field indicates the interrupt status for is calculated using the following formula:</p> $\text{NUM_FCH_PER_FCG} * n + m$ <p>where:</p> <p>n is the Fast Channel Group number</p> <p>m is the field offset</p> <p>If the calculated value is greater than the number of Fast Channels in the Mailbox the field is RES0</p> <p>To clear the interrupt software must acknowledge the last Transfer on the Fast Channel by reading the MFCW<n>_PAY register associated with the Channel.</p> <p>0b0</p> <p>Fast Channel N has an outstanding interrupt</p> <p>0b1</p> <p>Fast Channel has an outstanding interrupt</p> <p>Only bits NUM_FCH_PER_FCG-1:0 are implemented, with all unused bits being RES0</p>	0x00000000

Accessibility

Table 5-328: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(4 * n) + 0x0480	MBX_FCH_GRP<n>_INT_ST	None

5.5.2.14 MBX_IIDR, Mailbox Implementer Identification Register

This field provides information on the Implementer of the MHU

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FC8

Bit descriptions

Figure 5-162: MHUR.MBX_MBX_IIDR bit assignments

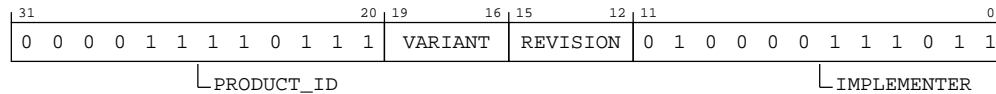


Table 5-329: MBX_IIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	PRODUCT_ID	Product ID of the MHU implementation 0b000011110111	0x0F7
[19:16]	VARIANT	Variant or Major revision of the MHU implementation	The reset value depends on the product version used. • 0x0 - r0
[15:12]	REVISION	Revision or minor version of the MHU implementation	The reset value depends on the product version used. • 0x0 - p0 • 0x1 - p1
[11:0]	IMPLEMENTER	Implementer ID Contains the JEP106 identification information as follows: <ul style="list-style-type: none"> 11:8 - JEP106 continuation code of implementer 7 - Always 0 6:0 - JEP106 identity code of implementer 0b010000111011	0x43B

Accessibility

Table 5-330: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FC8	MBX_IIDR	None

5.5.2.15 MBX_AIDR, Mailbox Architecture Identification Register

Provides information on the implemented MHU architecture

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FCC

Bit descriptions

Figure 5-163: MHUR.MBX_MBX_AIDR bit assignments

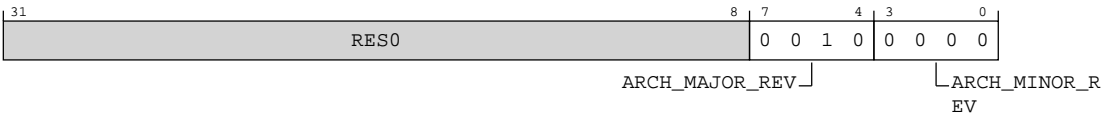


Table 5-331: MBX_AIDR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	ARCH_MAJOR_REV	MHU Architecture Major Revision 0b0010 MHUv3 All other values are Reserved	0b0010
[3:0]	ARCH_MINOR_REV	MHU Architecture Minor Revision 0b0000 Minor revision 0 of the major architecture All other values are Reserved	0b0000

Accessibility

Table 5-332: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FCC	MBX_AIDR	None

5.5.2.16 MBX_PIDR4, Mailbox Peripheral ID 4 Register

Returns byte[4] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FD0

Bit descriptions

Figure 5-164: MHUR.MBX_MBX_PIDR4 bit assignments

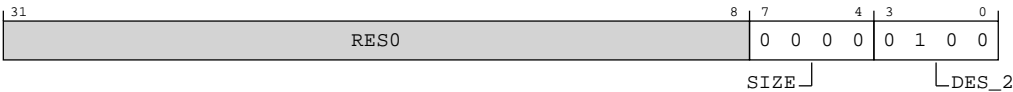


Table 5-333: MBX_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	Size of the component 0b0000 The size of the component must be identified using a combination of the following registers: <ul style="list-style-type: none">Peripheral ID 0-7 registersComponent ID 0-3 registers<x>_FEAT_SPT0 and <x>_FEAT_SPT1 registers	0b0000
[3:0]	DES_2	JEP106 continuation code 0b0100	0b0100

Accessibility

Table 5-334: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FD0	MBX_PIDR4	None

5.5.2.17 MBX_PIDR5, Mailbox Peripheral ID 5 Register

Returns byte[5] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FD4

Bit descriptions

Figure 5-165: MHUR.MBX_MBX_PIDR5 bit assignments



Table 5-335: MBX_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-336: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FD4	MBX_PIDR5	None

5.5.2.18 MBX_PIDR6, Mailbox Peripheral ID 6 Register

Returns byte[6] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FD8

Bit descriptions

Figure 5-166: MHUR.MBX_MBX_PIDR6 bit assignments

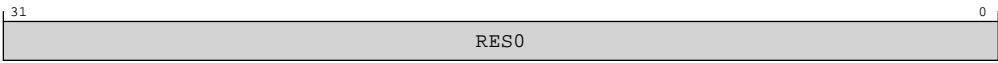


Table 5-337: MBX_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-338: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FD8	MBX_PIDR6	None

5.5.2.19 MBX_PIDR7, Mailbox Peripheral ID 7 Register

Returns byte[7] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FDC

Bit descriptions

Figure 5-167: MHUR.MBX_MBX_PIDR7 bit assignments

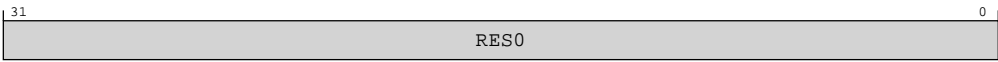


Table 5-339: MBX_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-340: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FDC	MBX_PIDR7	None

5.5.2.20 MBX_PIDR0, Mailbox Peripheral ID 0 Register

Returns byte[0] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FE0

Bit descriptions

Figure 5-168: MHUR.MBX_MBX_PIDR0 bit assignments



Table 5-341: MBX_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Bits 7:0 of the Part ID for the implementation of the MHU 0b11110111	0xF7

Accessibility

Table 5-342: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FE0	MBX_PIDR0	None

5.5.2.21 MBX_PIDR1, Mailbox Peripheral ID 1 Register

Returns byte[1] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FE4

Bit descriptions

Figure 5-169: MHUR.MBX_MBX_PIDR1 bit assignments

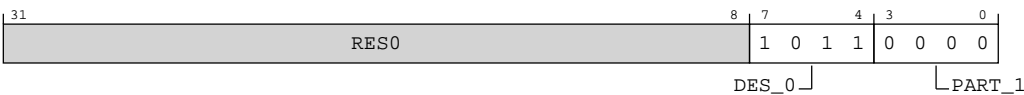


Table 5-343: MBX_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	DES_0	Bits 3:0 of the JEP106 identification code 0b1011	0b1011
[3:0]	PART_1	Bits 11:8 of the Part ID for the implementation of the MHU 0b0000	0b0000

Accessibility

Table 5-344: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FE4	MBX_PIDR1	None

5.5.2.22 MBX_PIDR2, Mailbox Peripheral ID 2 Register

Returns byte[2] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUR.MBX

Register offset
0x0FE8

Bit descriptions

Figure 5-170: MHUR.MBX_MBX_PIDR2 bit assignments

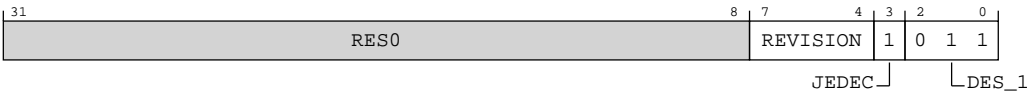


Table 5-345: MBX_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	The REVISION field is an incremental value starting at 0x0 for the first design of a component. The value is increased by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision.	The reset value depends on the product version used. <ul style="list-style-type: none">0x0 - r0p00x1 - r0p1
[3]	JEDEC	Must be 0b1 to indicate that a JEDEC-assigned value is used. 0b1	0b1
[2:0]	DES_1	JEP106 identification bits [6:4] 0b011	0b011

Accessibility

Table 5-346: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FE8	MBX_PIDR2	None

5.5.2.23 MBX_PIDR3, Mailbox Peripheral ID 3 Register

Returns byte[3] of the peripheral ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FEC

Bit descriptions

Figure 5-171: MHUR.MBX_MBX_PIDR3 bit assignments



Table 5-347: MBX_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	<p>The REVAND field indicates minor errata fixes specific to this design, for example metal fixes after implementation. Usually this field is zero.</p> <p>Together with PIDR2.REVISION, PIDR3.REVAND forms the revision number of the component.</p>	0x0
[3:0]	CMOD	<p>Customer Modified.</p> <p>If the component is reusable IP, the CMOD field indicates whether the customer has modified the behavior of the component.</p> <p>Arm recommends that the user or debugger reads the documentation for the component to determine the modifications that are made to the component.</p> <p>For any two components with the same values of the Peripheral ID 0-7 and Component ID 0-3 registers</p> <ul style="list-style-type: none">• If the value of the CMOD fields of both components equals zero, the components are identical• If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have been subjected to the same modifications.• If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same values of the Peripheral ID 0-7 and Component ID 0-3 registers	0x0

Accessibility

Table 5-348: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FEC	MBX_PIDR3	None

5.5.2.24 MBX_CIDR0, Mailbox Component ID 0 Register

Returns byte[0] of the component ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FF0

Bit descriptions

Figure 5-172: MHUR.MBX_MBX_CIDR0 bit assignments



Table 5-349: MBX_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble, segment 0 0b00001101	0x0D

Accessibility

Table 5-350: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FF0	MBX_CIDR0	None

5.5.2.25 MBX_CIDR1, Mailbox Component ID 1 Register

Returns byte[1] of the component ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0x0FF4

Bit descriptions

Figure 5-173: MHUR.MBX_MBX_CIDR1 bit assignments

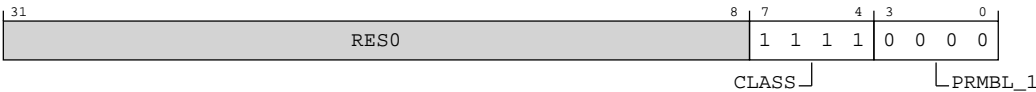


Table 5-351: MBX_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component Class 0b1111 CoreLink, PrimeCell, or system component with no standardized register layout, for backwards compatibility.	0b1111
[3:0]	PRMBL_1	Preamble, segment 1 0b0000	0b0000

Accessibility

Table 5-352: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FF4	MBX_CIDR1	None

5.5.2.26 MBX_CIDR2, Mailbox Component ID 2 Register

Returns byte[2] of the component ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
MHUR.MBX

Register offset
0x0FF8

Bit descriptions

Figure 5-174: MHUR.MBX_MBX_CIDR2 bit assignments



Table 5-353: MBX_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble, segment 2 0b00000101	0x05

Accessibility

Table 5-354: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FF8	MBX_CIDR2	None

5.5.2.27 MBX_CIDR3, Mailbox Component ID 3 Register

Returns byte[3] of the component ID for Mailbox page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUR.MBX

Register offset
0x0FFC

Bit descriptions

Figure 5-175: MHUR.MBX_MBX_CIDR3 bit assignments

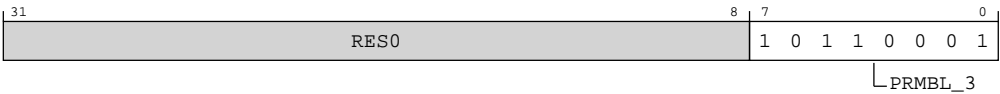


Table 5-355: MBX_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble, segment 3 0b10110001	0xB1

Accessibility

Table 5-356: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0x0FFC	MBX_CIDR3	None

5.5.2.28 MDBCW<n>_ST, Mailbox Doorbell Channel Window <n> Status Register, n = 0 - 127

Returns doorbell channel flags

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MDBCW<n>_ST are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(32 * n) + 0x1000

Bit descriptions

Figure 5-176: MHUR.MBX_MDBCW<n>_ST bit assignments

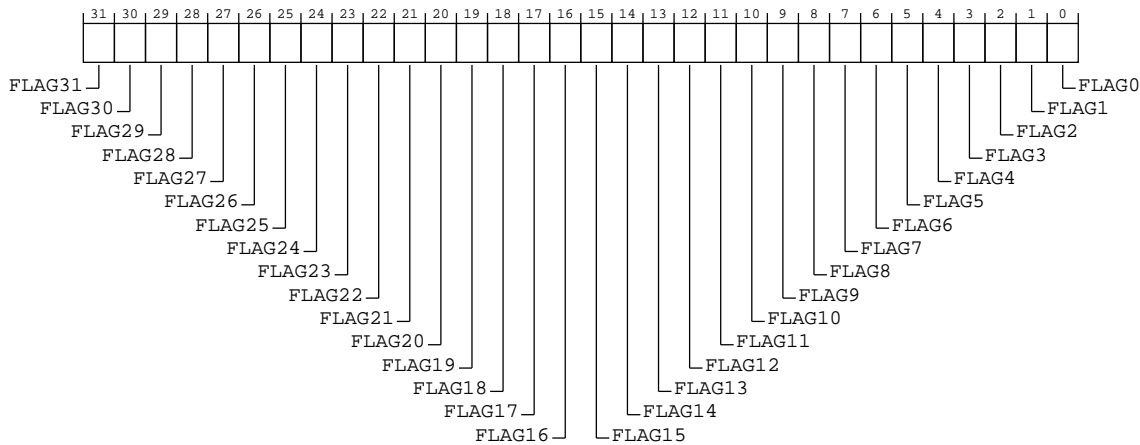


Table 5-357: MDBCW<n>_ST bit descriptions

Bits	Name	Description	Reset
[31:0]	FLAG<x>	Indicates the status of Flag bit <x> of the DBCH 0b0 Flag<x> bit is not set 0b1 Flag<x> bit is set	0x00000000

Accessibility

Table 5-358: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(32 * n) + 0x1000	MDBCW<n>_ST	None

5.5.2.29 MDBCW<n>_ST_MSK, Mailbox Doorbell Channel Window <n> Status Masked Register, n = 0 - 127

Returns doorbell channel flags after combining with the mask

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MDBCW<n>_ST_MSK are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(32 * n) + 0x1004$

Bit descriptions

Figure 5-177: MHUR.MBX_MDBCW<n>_ST_MSK bit assignments

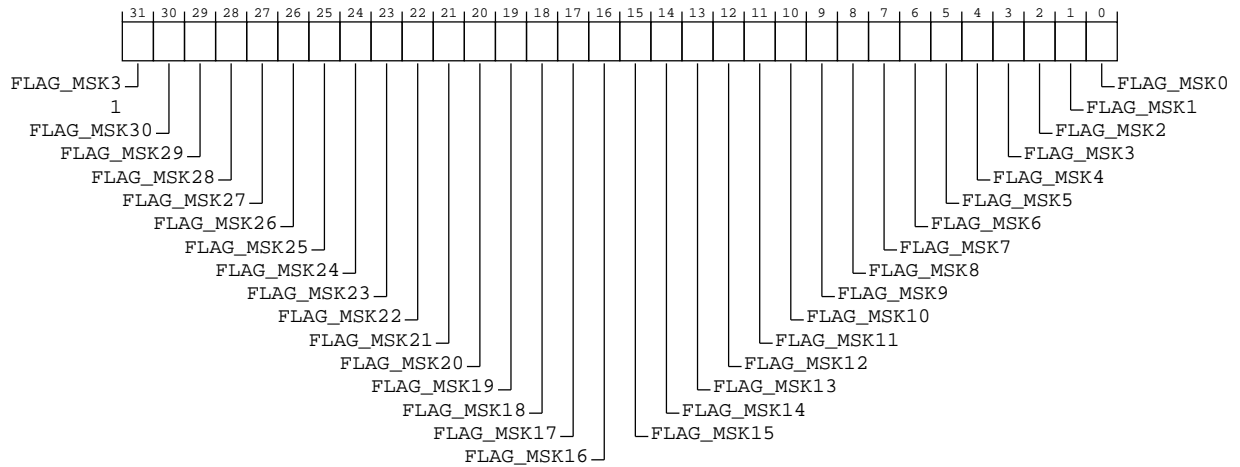


Table 5-359: MDBCW<n>_ST_MSK bit descriptions

Bits	Name	Description	Reset
[31:0]	FLAG_MSK<x>	Indicates the status of Flag bit <x> after the DBCH Mask has been applied for the DBCH 0b0 Flag Masked<x> bit is not set 0b1 Flag Masked<x> bit is set	0x00000000

Accessibility

Table 5-360: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	$(32 * n) + 0x1004$	MDBCW<n>_ST_MSK	None

5.5.2.30 MDBCW<n>_CLR, Mailbox Doorbell Channel Window <n> Clear Register, n = 0 - 127

Allows clearing doorbell channel flags

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MDBCW<n>_CLR are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(32 * n) + 0x1008$

Bit descriptions

Figure 5-178: MHUR.MBX_MDBCW<n>_CLR bit assignments

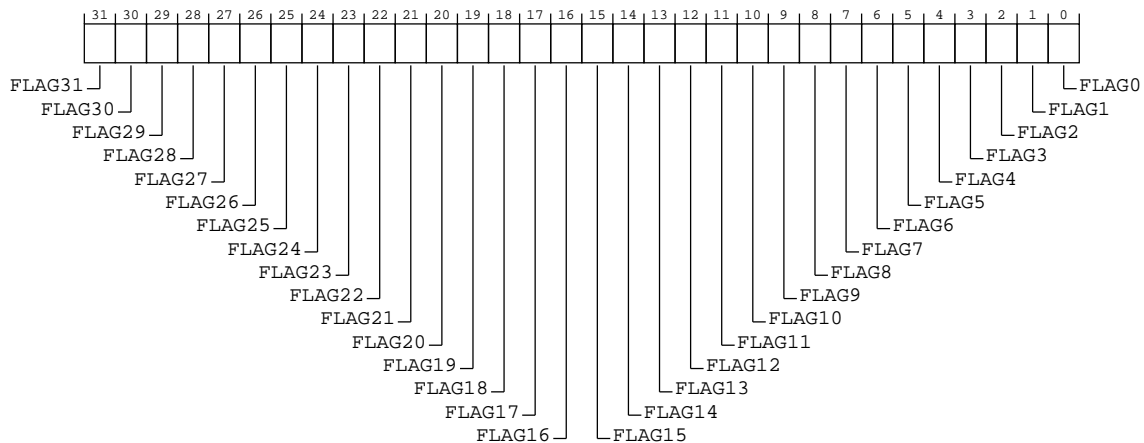


Table 5-361: MDBCW<n>_CLR bit descriptions

Bits	Name	Description	Reset
[31:0]	FLAG<x>	Writes of 1 cause the associated bit in the MDBCW<n>_ST registers to be set to 0. Writing 0 has no effect on the value of MDBCW<n>_ST. 0b0 No effect 0b1 Sets the associated bit in the MDBCW<n>_ST register to 0 Field always reads as 0	32 {x}

Accessibility

Table 5-362: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	$(32 * n) + 0x1008$	MDBCW<n>_CLR	None

5.5.2.31 MDBCW<n>_MSK_ST, Mailbox Doorbell Channel Window <n> Mask Status Register, n = 0 - 127

Returns doorbell channel mask

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MDBCW<n>_MSK_ST are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(32 * n) + 0x1010$

Bit descriptions

Figure 5-179: MHUR.MBX_MDBCW<n>_MSK_ST bit assignments

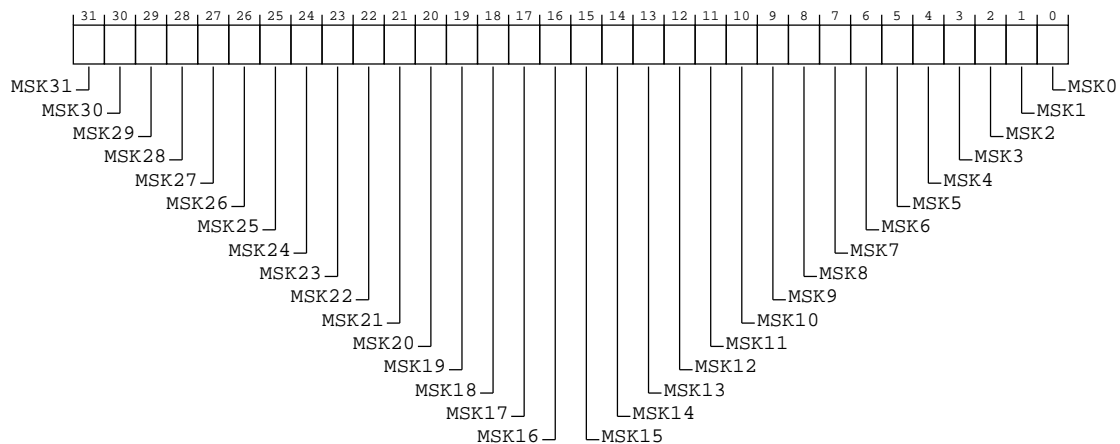


Table 5-363: MDBCW<n>_MSK_ST bit descriptions

Bits	Name	Description	Reset
[31:0]	MSK<x>	Indicates the status of Mask<x> of the DBCH mask 0b0 Mask<x> bit is not set 0b1 Mask<x> bit is set	0x00000000

Accessibility

Table 5-364: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(32 * n) + 0x1010	MDBCW<n>_MSK_ST	None

5.5.2.32 MDBCW<n>_MSK_SET, Mailbox Doorbell Channel Window <n> Mask Set Register, n = 0 - 127

Allows setting doorbell channel mask

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MDBCW<n>_MSK_SET are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(32 * n) + 0x1014

Bit descriptions

Figure 5-180: MHUR.MBX_MDBCW<n>_MSK_SET bit assignments

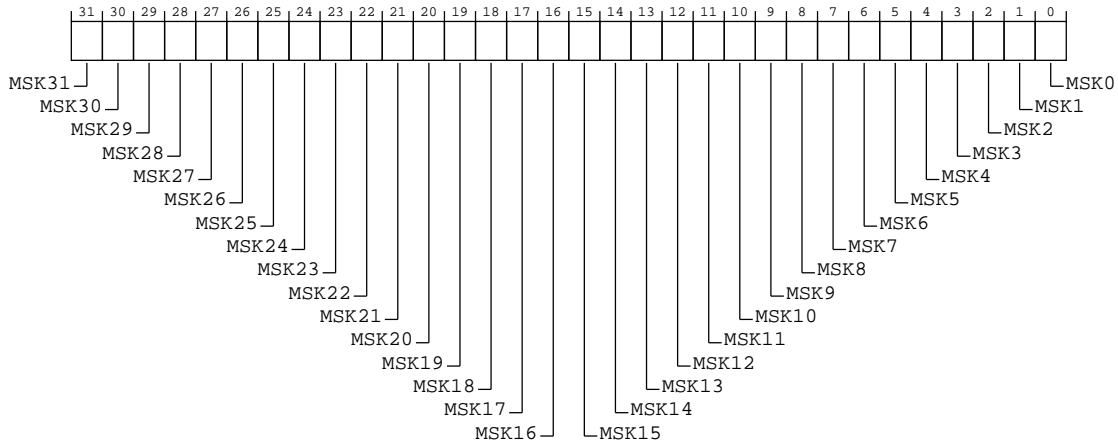


Table 5-365: MDBCW<n>_MSK_SET bit descriptions

Bits	Name	Description	Reset
[31:0]	MSK<x>	<p>Writes of 1 cause the associated bit in the MDBCW<n>_MSK_ST registers to be set to 1. Writing 0 has no effect on the value of MDBCW<n>_MSK_ST.</p> <p>0b0 No effect</p> <p>0b1 Set associated bit in MDBCW<n>_MSK_ST register to 1</p> <p>Field always reads as 0</p>	0x00000000

Accessibility

Table 5-366: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(32 * n) + 0x1014	MDBCW<n>_MSK_SET	None

5.5.2.33 MDBCW<n>_MSK_CLR, Mailbox Doorbell Channel Window <n> Mask Clear Register, n = 0 - 127

Allows clearing doorbell channel mask

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MDBCW<n>_MSK_CLR are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(32 * n) + 0x1018$

Bit descriptions

Figure 5-181: MHUR.MBX_MDBCW<n>_MSK_CLR bit assignments

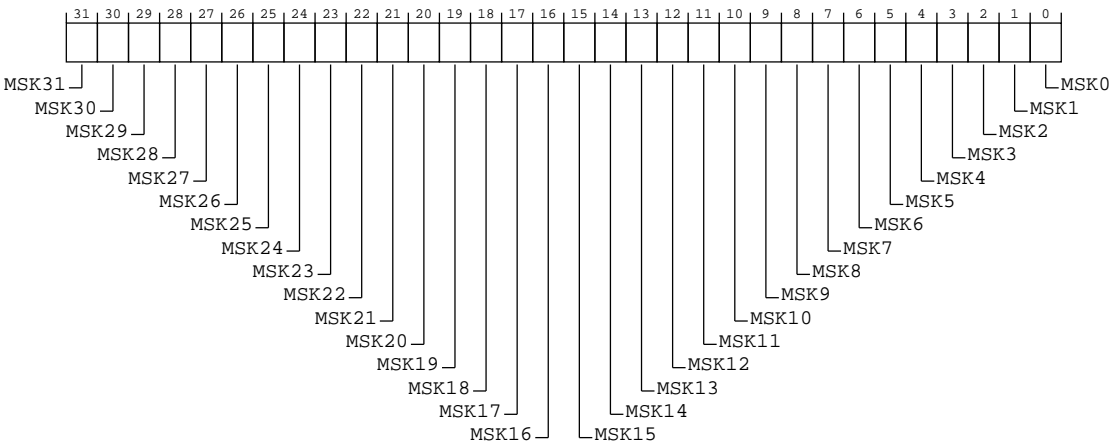


Table 5-367: MDBCW<n>_MSK_CLR bit descriptions

Bits	Name	Description	Reset
[31:0]	MSK<x>	Writes of 1 cause the associated bit in the MDBCW<n>_MSK_ST registers to be set to 0. Writing 0 has no effect on the value of MDBCW<n>_MSK_ST. 0b0 No effect 0b1 Set associated bit in MDBCW<n>_MSK_ST register to 0 Field always reads as 0	0x00000000

Accessibility

Table 5-368: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	$(32 * n) + 0x1018$	MDBCW<n>_MSK_CLR	None

5.5.2.34 MDBCW<n>_CTRL, Mailbox Doorbell Channel Window <n> Control Register, n = 0 - 127

This register contains control bits for doorbell channels

Configurations

This register is present only when DBE is implemented. Otherwise, direct accesses to MDBCW<n>_CTRL are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(32 * n) + 0x101C

Bit descriptions

Figure 5-182: MHUR.MBX_MDBCW<n>_CTRL bit assignments

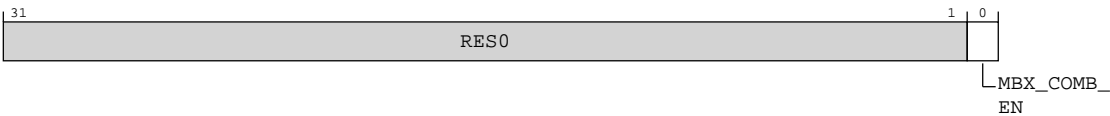


Table 5-369: MDBCW<n>_CTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	MBX_COMB_EN	Controls whether events from this Doorbell Channel contribute to the Mailbox Combined interrupt 0b0 Doorbell Channel does not contribute to the Mailbox Combined interrupt 0b1 Doorbell Channel contributes to the Mailbox Combined interrupt	0b1

Accessibility

Table 5-370: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(32 * n) + 0x101C	MDBCW<n>_CTRL	None

5.5.2.35 MFFCW<n>_PAY64, Mailbox FIFO Channel Window <n> Payload Register (64bit access), n = 0 - 63

A 64bit access to the MFFCW<n>_PAY register.

An access must be aligned to any 8bit boundary in the MFFCW<n>_PAY register, otherwise it is an unsupported access. It is implementation-defined whether the access is treated as RAZ/WI or modified to be an aligned access.

- 64bit accesses are only supported, if MBX_FFCH_CFG0.M64BA_SPT is set to 0b1.
- If MBX_FFCH_CFG0.M64BA_SPT is set to 0b0, 64bit accesses are not supported. It is implementation-defined whether unsupported accesses are treated as RAZ/WI or modified to a supported size.

MFFCW<n>_PAY register occupies offsets 0x00-0x07 in the Mailbox FIFO Channel Window <n>.

A read of this register reads up to eight bytes starting at the head of the FIFO, if the FIFO is not empty otherwise an IMPDEF value is returned.

If the MFFCW<n>_CTRL.RA_EN field is set to 0b1, the bytes read from the FIFO is also popped from the FIFO.

On a read of this register, the values of the data flags, associated with the bytes read from the FIFO are stored in the Flag History Buffer.

Configurations

This register is present only when FE is implemented and M64BA_SPT. Otherwise, direct accesses to MFFCW<n>_PAY64 are RAZ/WI.

Attributes

Width

64

Component

MHUR.MBX

Register offset

$(64 * n) + 0x2000$

Bit descriptions

A 64bit read access, aligned to a 64bit boundary, when the MHU implements 64bit accesses to the register

Figure 5-183: MFFCW<n>_PAY64_MFFCW<n>_PAY64 bit assignments

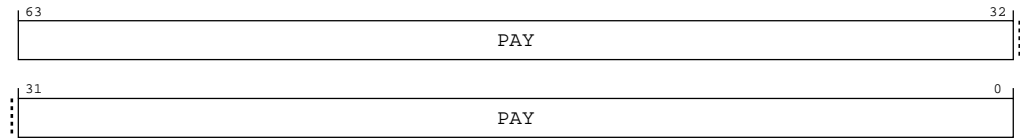


Table 5-371: MFFCW<n>_PAY64 bit descriptions

Bits	Name	Description	Reset
[63:0]	PAY	<p>Payload read from FIFO.</p> <p>Eight bytes are read from the FIFO, starting from the head of the FIFO.</p> <p>A byte read from the FIFO can be either valid or invalid. A valid byte has an 8bit data value and data flags associated with it. An invalid byte has an IMPDEF data value and no data flags associated with it.</p> <p>The data values of all bytes read from the FIFO, are arrange in the PAY field depending on the value of the MFFCW<n>_CTRL.MSBF field and the byte number.</p> <p>MFFCW<n>_CTRL.MSBF == 0b0</p> <p>Bytes are arranged in ascending order starting with the first byte read from the FIFO in the LSB of the PAY field and the last byte read from the FIFO in the MSB of the PAY field.</p> <p>MFFCW<n>_CTRL.MSBF == 0b1</p> <p>Bytes are arranged in ascending order starting with the first byte read from the FIFO in the MSB of the PAY field and the last byte read from the FIFO in the LSB of the PAY field.</p> <p>The data flags of each byte are stored in the Flag History Buffer(FHB) starting with the first byte read in the entry 0 of the FHB, with all other entries being marked as invalid.</p> <p>If MFFCW<n>_CTRL.RA_EN is set to 0b1, when a byte is read from the FIFO:</p> <ul style="list-style-type: none"> IT is also removed from the FIFO. A Transfer Acknowledge event is generated if that byte was associated with the ACK and EOT data flags. <p>Otherwise the byte still remains in the FIFO and a subsequent read of the same size would return the same bytes.</p>	See description.

Accessibility

Table 5-372: Accessibility

Component	Offset	Range
MFFCW<n>_PAY64	[(64 * n) + 0x2000]	63:0

5.5.2.36 MFFCW<n>_PAY32, Mailbox FIFO Channel Window <n> Payload Register (32bit access), n = 0 - 63

A 32bit access to the MFFCW<n>_PAY register.

Accesses must be aligned to any 8bit boundary within the MFFCW<n>_PAY register, otherwise it is an unsupported access and it is IMPDEF whether the access is treated as RAZ/WI or modified to be an aligned access.

32bit accesses are only supported, if MBX_FFCH_CFG0.M32BA_SPT is set to 0b1, otherwise it is an unsupported access and it is IMPDEF whether the access is treated as RAZ/WI or modified to a supported size.

If M64BA_SPT is set to 1 the MFFCW<n>_PAY register occupies offsets 0x00-0x07 within the Mailbox FIFO Channel Window <n>.

If M64BA_SPT is set to 0 the MFFCW<n>_PAY register occupies offsets 0x00-0x03 and offsets 0x04-0x07 are reserved, within the Mailbox FIFO Channel Window <n>.

A read of this register reads up to four bytes starting at the head of the FIFO, if the FIFO is not empty otherwise an IMPDEF value is returned.

If the MFFCW<n>_CTRL.RA_EN field is set to 0b1, the bytes read from the FIFO is also popped from the FIFO.

On a read of this register, the values of the data flags, associated with the bytes read from the FIFO are stored in the Flag History Buffer.

Configurations

This register is present only when FE is implemented and M32BA_SPT. Otherwise, direct accesses to MFFCW<n>_PAY32 are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offsets (2)

$(64 * n) + 0x2000, (64 * n) + 0x2004$

Bit descriptions

A 32bit aligned read access

Figure 5-184: MFFCW<n>_PAY32_MFFCW<n>_PAY32 bit assignments



Table 5-373: MFFCW<n>_PAY32 bit descriptions

Bits	Name	Description	Reset
[31:0]	PAY	<p>Payload read from FIFO</p> <p>Four bytes are read from the FIFO, starting from the head of the FIFO.</p> <p>A byte read from the FIFO can be either valid or invalid. A valid byte has an 8bit data value and data flags associated with it. An invalid byte has an IMPDEF data value and no data flags associated with it.</p> <p>The data values of all bytes read from the FIFO, are arrange in the PAY field depending on the value of the MFFCW<n>_CTRL.MSBF field and the byte number.</p> <p>MFFCW<n>_CTRL.MSBF == 0b0</p> <p>Bytes are arranged in ascending order starting with the first byte read from the FIFO in the LSB of the PAY field and the last byte read from the FIFO in the MSB of the PAY field.</p> <p>MFFCW<n>_CTRL.MSBF == 0b1</p> <p>Bytes are arranged in ascending order starting with the first byte read from the FIFO in the MSB of the PAY field and the last byte read from the FIFO in the LSB of the PAY field.</p> <p>The data flags of each byte are stored in the Flag History Buffer(FHB) starting with the first byte read in the entry 0 of the FHB, with all other entries being marked as invalid.</p> <p>If MFFCW<n>_CTRL.RA_EN is set to 0b1, when a byte is read from the FIFO:</p> <ul style="list-style-type: none"> it is also removed from the FIFO and a Transfer Acknowledge event is generated if that byte was associated with the ACK and EOT data flags <p>Otherwise the byte still remains in the FIFO and a subsequent read of the same size would return the same bytes.</p>	See description.

Accessibility

Table 5-374: Accessibility

Component	Offset	Range
MFFCW<n>_PAY32	[(64 * n) + 0x2000]	63:0

Component	Offset	Range
MFFCW<n>_PAY32	[(64 * n) + 0x2004]	63:0

5.5.2.37 MFFCW<n>_FLG64, Mailbox FIFO Channel Window <n> Flag Register (64bit access), n = 0 - 63

A 64bit access to the MFFCW<n>_FLG register.

An access must be aligned to any 64bit boundary in the MFFCW<n>_FLG register, otherwise the access is treated as **RAZ/WI**.

- 64bit accesses are only supported if MBX_FFCH_CFG0.M64BA_SPT is set to 0b1.
- If MBX_FFCH_CFG0.M64BA_SPT is set to 0b1, 64bit accesses are not supported. It is implementation-defined whether an unsupported access is treated as **RAZ/WI** or modified to a supported size.

The MFFCW<n>_FLG register occupies offsets 0x08-0x0F in the Mailbox FIFO Channel Window <n>.

A read of this register returns:

- The contents of the Flag History Buffer
- Current fill level of the FIFO

This register is expected to be read after a read of the MFFCW<n>_PAY register to get the data flags associated with the bytes read from the FIFO. The read of this register must be of the same size as the read of the MFFCW<n>_PAY register, otherwise it can lead to loss or corruption of information.

Configurations

This register is present only when FE is implemented and 64 bit access are supported. Otherwise, direct accesses to MFFCW<n>_FLG64 are RAZ/WI.

Attributes

Width

64

Component

MHUR.MBX

Register offset

$(64 * n) + 0x2008$

Bit descriptions

Figure 5-185: MFFCW<n>_FLG64_MFFCW<n>_FLG64 bit assignments

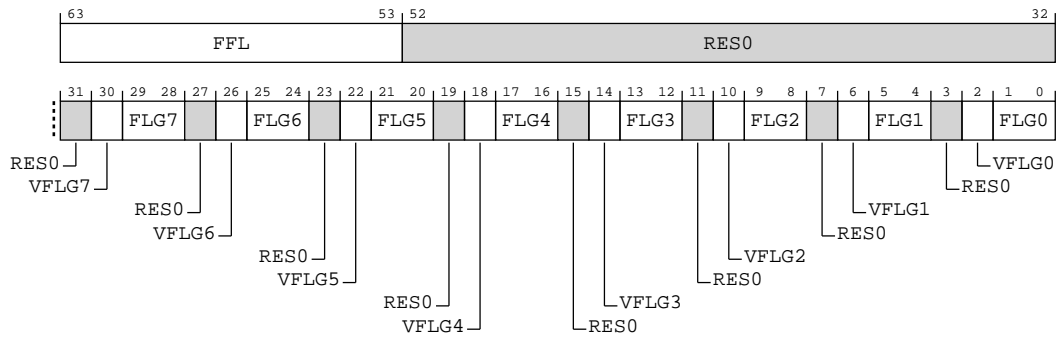


Table 5-376: MFFCW<n>_FLG64 bit descriptions

Bits	Name	Description	Reset
[63:53]	FFL	FIFO Fill Level Indicates the number of bytes containing valid data in the FIFO 0b000000000000 All bytes in the FIFO are invalid The maximum value returned is never greater than the FIFO Depth	11{x}
[52:31, 27, 23, 19, 15, 11, 7, 3]	RES0	Reserved	RES0
[30, 26, 22, 18, 14, 10, 6, 2]	VFLG<m>	Valid Flag <m> Indicates whether FLG<m> field contains valid data or not. 0b0 FLG<m> is not valid 0b1 FLG<m> is valid	0x00

Bits	Name	Description	Reset
[29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0]	FLG<m>	<p>Flag <m></p> <p>Provides the data flags, except for the ACK flag, held in FHB entry 0 to 7 and whether the data flags are valid or invalid.</p> <p>The association of the FLG<m> field to the FHB entries depends on the value of the MFFCW<n>_CTRL.MSBF field when the read of the MFFCW<n>_FLG register occurs</p> <p>MFFCW<n>_CTRL.MSBF == 0b0</p> <p>FLG0 is associated with FHB entry 0 and FLG7 is associated with FHB entry 7</p> <p>MFFCW<n>_CTRL.MSBF == 0b1</p> <p>FLG0 is associated with FHB entry 7 and FLG7 is associated with FHB entry 0</p> <p>The legal values of the FLG<m> field, when VFLG<m> is set to 0b1 are:</p> <p>0b00</p> <p>Payload</p> <p>Neither the first or last byte of a Transfer</p> <p>0b01</p> <p>Start Byte</p> <p>First byte of a Transfer</p> <p>0b10</p> <p>End Byte</p> <p>Last byte of a Transfer</p> <p>0b11</p> <p>Start and End Byte</p> <p>First and last byte of a Transfer</p> <p>The value of the FLG<m> field is only valid if VFLG<m> is set to 0b1, otherwise the value of this field must be ignored by software.</p>	16 {x}

Accessibility

Table 5-377: Accessibility

Component	Offset	Range
MFFCW<n>_FLG64	[(64 * n) + 0x2008]	63:0

5.5.2.38 MFFCW<n>_FLG32, MailboxPostbox FIFO Channel Window <n> Flag Register (32bit access), n = 0 - 63

A 32bit access to the MFFCW<n>_FLG register.

An access must be aligned to any 32bit boundary in the MFFCW<n>_FLG register, otherwise the access is treated as **RAZ/WI**.

- 32bit accesses are only supported, if MBX_FFCH_CFG0.M32BA_SPT is set to 0b1.
- If MBX_FFCH_CFG0.M32BA_SPT is set to 0b1, 32bit accesses are not supported. It is implementation-defined whether the access is treated as **RAZ/WI** or modified to a supported size.

The number of MFFCW<n>_FLG32 registers depends on the MBX_FFCH_CFG0.M64BA_SPT register field:

- If M64BA_SPT is set to 1, there are two MFFCW<n>_FLG registers, which occupy offsets 0x08-0x0F in the Mailbox FIFO Channel Window <n>.
- If M64BA_SPT is set to, 0 the MFFCW<n>_FLG register occupies offsets 0x08-0x0B and offsets 0x0C-0x0F are reserved, in the Mailbox FIFO Channel Window <n>.

A read of this register returns:

- The contents of the Flag History Buffer
- Current fill level of the FIFO

This register is expected to be read after a read of the MFFCW<n>_PAY register to get the data flags associated with the bytes read from the FIFO. The read of this register must be of the same size as the read of the MFFCW<n>_PAY register, otherwise it can lead to loss or corruption of information.

Configurations

This register is present only when FE is implemented and M32BA_SPT. Otherwise, direct accesses to MFFCW<n>_FLG32 are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offsets (2)

$(64 * n) + 0x2008, (64 * n) + 0x200C$

Bit descriptions

Figure 5-186: MFFCW<n>_FLG32_MFFCW<n>_FLG32 bit assignments

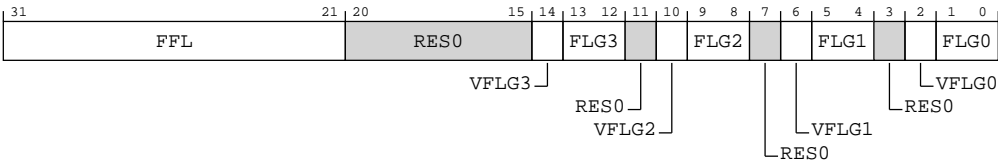


Table 5-378: MFFCW<n>_FLG32 bit descriptions

Bits	Name	Description	Reset
[31:21]	FFL	FIFO Fill Level Indicates the number of bytes containing valid data in the FIFO 0b000000000000 All bytes in the FIFO are invalid The maximum value returned is never greater than the FIFO Depth	11{x}
[20:15, 11, 7, 3]	RES0	Reserved	RES0
[14, 10, 6, 2]	VFLG<m>	Valid Flag <m> Indicates whether FLG<m> field contains valid data or not. 0b0 FLG<m> is not valid 0b1 FLG<m> is valid	0b0000

Bits	Name	Description	Reset
[13:12, 9:8, 5:4, 1:0]	FLG<m>	<p>Flag <m></p> <p>Provides the data flags, except for the ACK flag, held in FHB entry 0 to 3 and whether the data flags are valid or invalid.</p> <p>The association of the FLG<m> field to the FHB entries depends on the value of the MFFCW<n>_CTRL.MSBF field when the read of the MFFCW<n>_FLG register occurs</p> <p>MFFCW<n>_CTRL.MSBF == 0b0</p> <p>FLG0 is associated with FHB entry 0 and FLG3 is associated with FHB entry 3</p> <p>MFFCW<n>_CTRL.MSBF == 0b1</p> <p>FLG0 is associated with FHB entry 3 and FLG3 is associated with FHB entry 0</p> <p>The legal values of the FLG<m> field, when VFLG<m> is set to 0b1 are:</p> <p>0b00</p> <p>Payload</p> <p>Neither the first or last byte of a Transfer</p> <p>0b01</p> <p>Start Byte</p> <p>First byte of a Transfer</p> <p>0b10</p> <p>End Byte</p> <p>Last byte of a Transfer</p> <p>0b11</p> <p>Start and End Byte</p> <p>First and last byte of a Transfer</p> <p>The value of the FLG<m> field is only valid if VFLG<m> is set to 0b1, otherwise the value of this field must be ignored by software.</p>	8{x}

Accessibility

Table 5-379: Accessibility

Component	Offset	Range
MFFCW<n>_FLG32	[(64 * n) + 0x2008]	63:0

Component	Offset	Range
MFFCW<n>_FLG32	[(64 * n) + 0x200C]	63:0

5.5.2.39 MFFCW<n>_INT_ST, Mailbox FIFO Channel Window <n> Interrupt Status Register, n = 0 - 63

Returns FIFO channel interrupt status

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MFFCW<n>_INT_ST are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(64 * n) + 0x2010

Bit descriptions

Shows the status of the events of the DBCH

Figure 5-187: MHUR.MBX_MFFCW<n>_INT_ST bit assignments

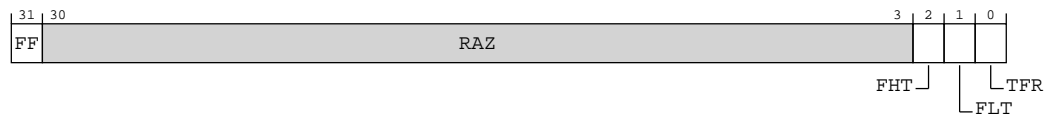


Table 5-381: MFFCW<n>_INT_ST bit descriptions

Bits	Name	Description	Reset
[31]	FF	0b0 No FIFO flush event has occurred 0b1 FIFO flush event has occurred	0b0
[30:3]	RAZ	Reserved	RAZ
[2]	FHT	0b0 No FIFO High Tidemark event has occurred 0b1 FIFO High Tidemark event has occurred	0b0
[1]	FLT	0b0 No FIFO Low Tidemark event has occurred 0b1 FIFO Low Tidemark event has occurred	0b0

Bits	Name	Description	Reset
[0]	TFR	0b0 No Transfer event 0b1 Transfer event	0b0

Accessibility

Table 5-382: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(64 * n) + 0x2010	MFFCW<n>_INT_ST	None

5.5.2.40 MFFCW<n>_INT_CLR, Mailbox FIFO Channel Window <n> Interrupt Clear Register, n = 0 - 63

Register for clearing FIFO channel interrupt status

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MFFCW<n>_INT_CLR are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(64 * n) + 0x2014

Bit descriptions

Shows the status of the events of the DBCH

Figure 5-188: MHUR.MBX_MFFCW<n>_INT_CLR bit assignments

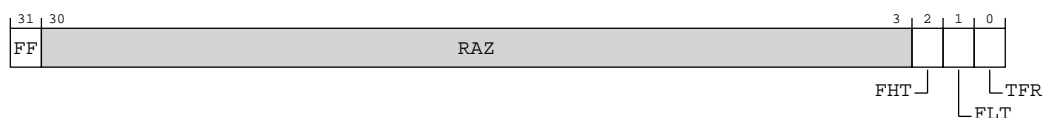


Table 5-383: MFFCW<n>_INT_CLR bit descriptions

Bits	Name	Description	Reset
[31]	FF	0b0 No effect 0b1 Clear FIFO flush event	x
[30:3]	RAZ	Reserved	RAZ
[2]	FHT	0b0 No effect 0b1 Clear FIFO High Tidemark event	0b0
[1]	FLT	0b0 No effect 0b1 Clear FIFO Low Tidemark event	0b0
[0]	TFR	0b0 No effect 0b1 Clear Transfer event	0b0

Accessibility

Table 5-384: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(64 * n) + 0x2014	MFFCW<n>_INT_CLR	None

5.5.2.41 MFFCW<n>_INT_EN, Mailbox FIFO Channel Window <n> Interrupt Enable Register, n = 0 - 63

Register for configuring FIFO channel interrupt enables

This register shows the status of FIFO channel events.

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MFFCW<n>_INT_EN are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset
(64 * n) + 0x2018

Bit descriptions

Figure 5-189: MHUR.MBX_MFFCW<n>_INT_EN bit assignments

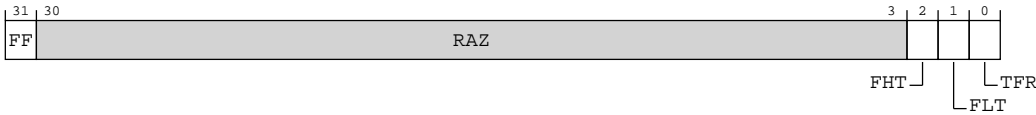


Table 5-385: MFFCW<n>_INT_EN bit descriptions

Bits	Name	Description	Reset
[31]	FF	0b0 FIFO flush events do not generate an interrupt 0b1 FIFO flush events generate an interrupt	0b1
[30:3]	RAZ	Reserved	RAZ
[2]	FHT	0b0 FIFO High Tidemark events do not generate an interrupt 0b1 FIFO High Tidemark events generate an interrupt	0b0
[1]	FLT	0b0 FIFO Low Tidemark events do not generate an interrupt 0b1 FIFO Low Tidemark events generate an interrupt	0b0
[0]	TFR	0b0 Transfer events do not generate an interrupt 0b1 Transfer events generate interrupts	0b1

Accessibility

Table 5-386: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(64 * n) + 0x2018	MFFCW<n>_INT_EN	None

5.5.2.42 MFFCW<n>_CTRL, Mailbox FIFO Channel Window <n> Control Register,
n = 0 - 63

This register contains control bits for FIFO channels

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MFFCW<n>_CTRL are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(64 * n) + 0x2020

Bit descriptions

Figure 5-190: MHUR.MBX_MFFCW<n>_CTRL bit assignments

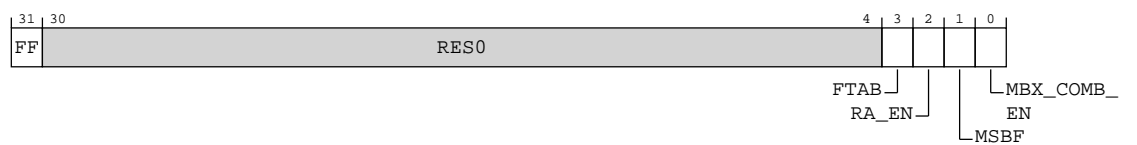


Table 5-387: MFFCW<n>_CTRL bit descriptions

Bits	Name	Description	Reset
[31]	FF	FIFO Flush Request a flush of the FFCH 0b0 No request to flush the FIFO 0b1 Request to flush the FIFO	0b0
[30:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3]	FTAB	<p>Future Transfer Auto Buffering</p> <p>0b0</p> <p>Future Transfer Auto Buffering is not enabled</p> <p>The value of flags associated with bytes read from the FIFO have no effect on the number of bytes read from the FIFO</p> <p>A request to pop bytes from the FIFO is not affected</p> <p>0b1</p> <p>Future Transfer Auto Buffering is enabled</p> <p>When a read of the MFFCW<n>_PAY registers with an access size greater than 1 occurs, any bytes read from the FIFO after the first byte with the EOT field set to 0b1 has been detected, are:</p> <ul style="list-style-type: none"> • Not read or popped from the FIFO • the bytes are set to an UNKNOWN value in the read data response • entries in the Flag History Buffer are set to invalid <p>Only bytes up to the first byte read from the FIFO with the EOT field set to 0b1 are popped from the FIFO</p> <p>Must be used with the RA_EN field set to 0b1, otherwise the field is considered to be 0b0</p>	0b0
[2]	RA_EN	<p>Controls whether Read to acknowledge is enabled</p> <p>0b0</p> <p>Read acknowledge is not enabled</p> <p>Bytes are popped from the FIFO by writing to MFFCW<n>_PAY register.</p> <p>The number of bytes popped from the FIFO is determined by the:</p> <ul style="list-style-type: none"> • Size of the access to the MFFCW<n>_PAY • Lowest offset within the MFFCW<n>_PAY register the access targets. • Number of valid bytes in the FIFO <p>0b1</p> <p>Read acknowledge is enabled</p> <p>Bytes are popped from the FIFO by reading the MFFCW<n>_PAY register.</p> <p>The number of bytes popped from the FIFO is determined by the:</p> <ul style="list-style-type: none"> • Size of the access • FTAB field value • Value of the EOT flag for a byte which is popped from the FIFO due to this read, when the FTAB field is set 0b1 • Lowest offset within the MFFCW<n>_PAY that is targeted by the read access • Number of valid bytes in the FIFO 	0b0

Bits	Name	Description	Reset
[1]	MSBF	<p>Most Significant Byte First</p> <p>Selects the order in which bytes are pushed onto the FIFO when multiple bytes are to be pushed due to a single write to the MFFCW<n>_PAY register</p> <p>0b0 Least Significant Byte first</p> <p>0b1 Most Significant Byte first</p> <p>FFCH are considered little endian and the LSB is the byte lowest offset within the access and the MSB is the highest byte offset within the access</p>	0b0
[0]	MBX_COMB_EN	<p>0b0 FIFO Channel does not contribute to the Mailbox Combined interrupt</p> <p>0b1 FIFO Channel contributes to the Mailbox Combined interrupt</p>	0b1

Accessibility

Table 5-388: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	$(64 * n) + 0x2020$	MFFCW<n>_CTRL	None

5.5.2.43 MFFCW<n>_ST, Mailbox FIFO Channel Window <n> Status Register, n = 0 - 63

Contains status information for FIFO channel

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MFFCW<n>_ST are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(64 * n) + 0x2024$

Bit descriptions

Figure 5-191: MHUR.MBX_MFFCW<n>_ST bit assignments



Table 5-389: MFFCW<n>_ST bit descriptions

Bits	Name	Description	Reset
[31]	FF	<p>FIFO Flush.</p> <p>Status of a flush of the FFCH.</p> <p>0b0</p> <p>The meaning of this value depends on the value of the FF field in the corresponding MFFCW<n>_CTRL register:</p> <ul style="list-style-type: none"> When MFFCW_CTRL<n>.FF is 0b0 - Receiver FIFO flush mechanism is idle When MFFCW_CTRL<n>.FF is 0b1 - Receiver FIFO flush mechanism is performing a flush <p>0b1</p> <p>FIFO flush complete</p>	0b0
[30:11]	RES0	Reserved	RES0
[10:0]	FFL	<p>FIFO Fill Level.</p> <p>Indicates the number of bytes containing valid data in the FIFO.</p> <p>The maximum value returned is never greater than the FIFO Depth.</p>	11 {x}

Accessibility

Table 5-390: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	(64 * n) + 0x2024	MFFCW<n>_ST	None

5.5.2.44 MFFCW<n>_FIFO_POP, Mailbox FIFO Channel Window <n> FIFO POP Register, n = 0 - 63

Register for popping bytes from a FIFO channel when writing to it

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MFFCW<n>_FIFO_POP are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

(64 * n) + 0x2028

Bit descriptions

Figure 5-192: MHUR.MBX_MFFCW<n>_FIFO_POP bit assignments

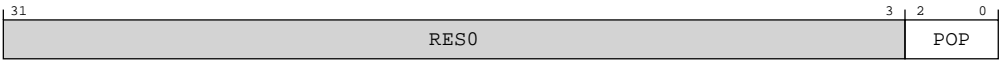


Table 5-391: MFFCW<n>_FIFO_POP bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2:0]	POP	<p>Number of bytes to pop from the FIFO. This is the maximum number of bytes popped from the FIFO. The number of bytes actually popped from the FIFO is as follows:</p> <ul style="list-style-type: none"> When MFFCW<n>_CTRL.RA_EN is set to 0b1 no bytes are popped from the FIFO. When MFFCW<n>_CTRL.RA_EN is set to 0b0 the number of bytes is the smallest of the following: <ul style="list-style-type: none"> Value written to this field. Number of valid bytes in the FIFO. <p>0b000 It is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>0b001 It is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>0b010 It is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>0b011 When M32BA_SPT, maximum of 4 bytes are popped from the FIFO.</p> <p>When !M32BA_SPT, it is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>0b100 It is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>0b101 It is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>0b110 It is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>0b111 When !M64BA_SPT, it is IMPDEF whether:</p> <ul style="list-style-type: none"> No bytes are popped from the FIFO The value is treated as another supported value <p>When M64BA_SPT, maximum of 8 bytes are popped from the FIFO.</p>	0b000

Accessibility

Table 5-392: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	$(64 * n) + 0x2028$	MFFCW<n>_FIFO_POP	None

5.5.2.45 MFFCW<n>_TIDE, Mailbox FIFO Channel Window <n> Tidemark Register, n = 0 - 63

Allows configuration of the low and high tidemark thresholds for the Receiver tidemark events

Configurations

This register is present only when FE is implemented. Otherwise, direct accesses to MFFCW<n>_TIDE are RAZ/WI.

Attributes

Width

32

Component

MHUR.MBX

Register offset

$(64 * n) + 0x202C$

Bit descriptions

Figure 5-193: EXT_MFFCW<n>_TIDE bit assignments

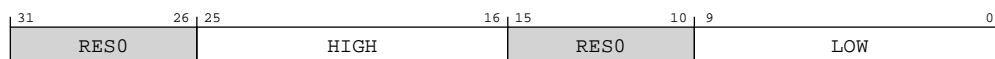


Table 5-393: MFFCW<n>_TIDE bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[25:16]	HIGH	<p>High Tide Mark</p> <p>Threshold value used in the generation of the Receiver FIFO High Tide event</p> <p>The event is generated when a push to the FIFO occurs, and the following are both true:</p> <ul style="list-style-type: none"> The FIFO fill level before the push was less than or equal to the value of this field The FIFO fill level after the push is greater than the value of this field <p>The upper and lower offset of this field depend on the configuration of the MHU.</p> <p>The upper offset of this field is equal to $\text{clog2}(\text{FFCH_DEPTH})+15$</p> <p>The lower offset of this field depends on the supported access sizes to MFFCW<n>_PAY register as follows:</p> <ul style="list-style-type: none"> When MBX_FFCH_CFG.M{8/16}BA_SPT are both 0b0 and MBX_FFCH_CFG.M32BA_SPT is 0b1, the lower offset is 18. <p>Any offsets above the upper offset or below the lower offset are RES0.</p> <p>If the upper offset is less than the lower offset the entire field is RES0.</p> <p>In all cases the value of this field includes the RES0 offsets for calculation in the Receiver High Tide event.</p>	See field description.
[15:10]	RES0	Reserved	RES0
[9:0]	LOW	<p>Low Tide Mark</p> <p>Threshold value used in the generation of the Receiver FIFO Low Tide event</p> <p>The event is generated when a pop to the FIFO occurs, and the following are both true:</p> <ul style="list-style-type: none"> The FIFO fill level before the pop was greater than the value of this field The FIFO fill level after the pop is less than or equal the value of this field <p>The upper and lower offset of this field depend on the configuration of the MHU.</p> <p>The upper offset of this field is equal to $\text{clog2}(\text{FFCH_DEPTH})-1$</p> <p>The lower offset of this field depends on the supported access sizes to MFFCW<n>_PAY register as follows:</p> <ul style="list-style-type: none"> When MBX_FFCH_CFG.M{8/16}BA_SPT are both 0b0 and MBX_FFCH_CFG.M32BA_SPT is 0b1, the lower offset is 2. <p>Any offsets above the upper offset or below the lower offset are RES0.</p> <p>If the upper offset is less than the lower offset the entire field is RES0.</p> <p>In all cases the value of this field includes the RES0 offsets for calculation in the Receiver Low Tide event.</p>	0b0000000000

5.5.2.46 MFCW<n>_PAY32, Mailbox Fast Channel Window <n> Payload 32bit Register, n = 0 - 1023

Access to payload of Fast Channel <n>

Arm recommends that accesses to these registers are atomic.
This is the 32bit version of the MFCW<n>_PAY registers

Configurations

This register is present only when FCE is implemented and FCH_WS == 0x20. Otherwise, direct accesses to MFCW<n>_PAY32 are RAZ/WI.

Attributes

Width
32

Component
MHUR.MBX

Register offsets
(4 * n) + 0x3000

Bit descriptions

Figure 5-194: MFCW<n>_PAY32_MFCW<n>_PAY32 bit assignments



Table 5-394: MFCW<n>_PAY32 bit descriptions

Bits	Name	Description	Reset
[31:0]	PAY	<div> <div> Payload for Channel <n></div> <div> A write to this register sets the value of the payload</div> <div> A read to this register: <ul style="list-style-type: none"> returns the current value of the payload acknowledges the Transfer clears the Transfer interrupt for the Channel, if it is set </div> </div>	32 { x }

Accessibility

Table 5-395: Accessibility

Component	Offset	Range
MFCW<n>_PAY32	[(4 * n) + 0x3000]	1023:0

5.5.2.47 MFCW<n>_PAY64, Mailbox Fast Channel Window <n> Payload 64bit Register, n = 0 - 511

Access to payload of Fast Channel <n>

Arm recommends that accesses to these registers are atomic.
This is the 64bit version of the MFCW<n>_PAY registers

Configurations

This register is present only when FCE is implemented and FCH_WS == 0x40. Otherwise, direct accesses to MFCW<n>_PAY64 are RAZ/WI.

Attributes

Width

64

Component

MHUR.MBX

Register offsets

$(8 * n) + 0x3000$

Bit descriptions

Figure 5-195: MFCW<n>_PAY64_MFCW<n>_PAY64 bit assignments

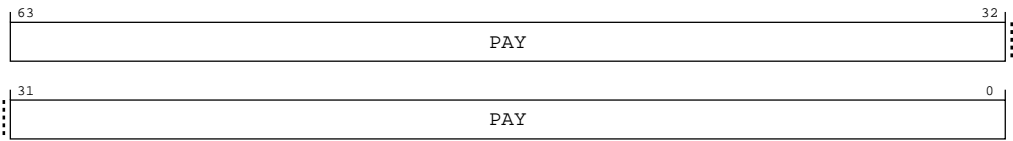


Table 5-396: MFCW<n>_PAY64 bit descriptions

Bits	Name	Description	Reset
[63:0]	PAY	<p>Payload for Channel <n></p> <p>A write to this register sets the value of the payload</p> <p>A read to this register:</p> <ul style="list-style-type: none">• returns the current value of the payload• acknowledges the Transfer• clears the Transfer interrupt for the Channel, if it is set	64 {x}

Accessibility

Table 5-397: Accessibility

Component	Offset	Range
MFCW<n>_PAY64	$[(8 * n) + 0x3000]$	511:0

5.5.2.48 MBX_FCTRL, Mailbox Feature Control Register

Controls non-architectural mailbox functionality

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.MBX

Register offset

0xF000

Bit descriptions

Figure 5-196: MHUR.MBX_MBX_FCTRL bit assignments

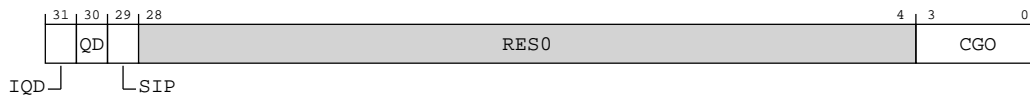


Table 5-398: MBX_FCTRL bit descriptions

Bits	Name	Description	Reset
[31]	IQD	Idle Q-Deny. Forces clock Q-channel to deny if all channels are not idle in the Receiver.	0b0
[30]	QD	Q-Deny. Forces clock Q-channel to always deny in the Receiver.	0b0
[29]	SIP	Scrub In Progress. Controls scrub of all RAMs in the Receiver. 0b0 Abort scrub, if written. Scrub not in progress, if read. 0b1 Start scrub, if written. Scrub in progress, if read.	0b0
[28]	RES0	Reserved	RES0
[27]	MSIL	MHU-Stream ID Limit. Limits number of AWID values used on ACE-Lite MHU-Stream manager interface. 0b0 Use all available AWID values as per protocol. 0b1 Use only AWID=0. This field must be programmed only out of reset and not changed during operation.	0b0
[26:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:0]	CGO	<p>Clock Gate Override. Disables architectural clock gating.</p> <p>The clock gate bit assignments are:</p> <ul style="list-style-type: none"> bit[3]: RAS register control bit[2]: FIFO channel control bit[1]: Fast channel control bit[0]: Doorbell channel control 	0b0000

Accessibility

Table 5-399: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0xF000	MBX_FCTRL	None

5.5.2.49 MBX_DB_ERRINS, Mailbox doorbell channel RAM ECC Error Insertion Register

Enables ECC error insertion in the Mailbox doorbell channel RAM. The bit descriptions for this register depend on whether the access is a write or a read.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.MBX

Register offset

0xF010

Bit descriptions

When the access is a write, the MBX_DB_ERRINS register has the following bit assignments.

Figure 5-197: MHUR.MBX_MBX_DB_ERRINS write bit assignments

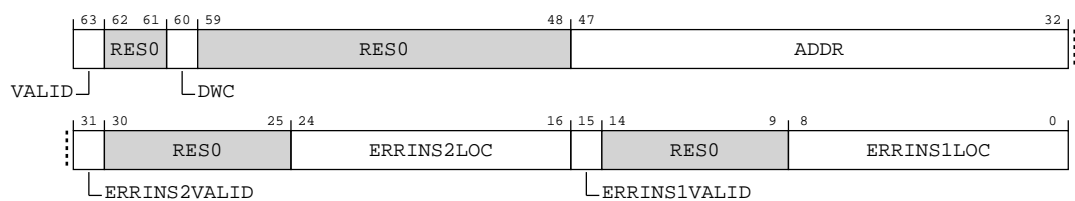


Table 5-400: MBX_DB_ERRINS write bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Writing 1'b1 triggers starting error insertion operation.	x
[62:61]	RES0	Reserved	RES0
[60]	DWC	Disable Write Check. If set, tests the ECC encoder.	x
[59:48]	RES0	Reserved	RES0
[47:32]	ADDR	RAM address that required error will be inserted at.	16 {x}
[31]	ERRINS2VALID	If set, enables insertion of second RAM error in bit location specified by ERRINS2LOC	x
[30:25]	RES0	Reserved	RES0
[24:16]	ERRINS2LOC	Defines bit location for second inserted RAM error	9 {x}
[15]	ERRINS1VALID	If set, enables insertion of first RAM error in bit location specified by ERRINS1LOC	x
[14:9]	RES0	Reserved	RES0
[8:0]	ERRINS1LOC	Defines bit location for first inserted RAM error	9 {x}

When the access is a read, the MBX_DB_ERRINS register has the following bit assignments.

Figure 5-198: MHUR.MBX_MBX_DB_ERRINS read bit assignments

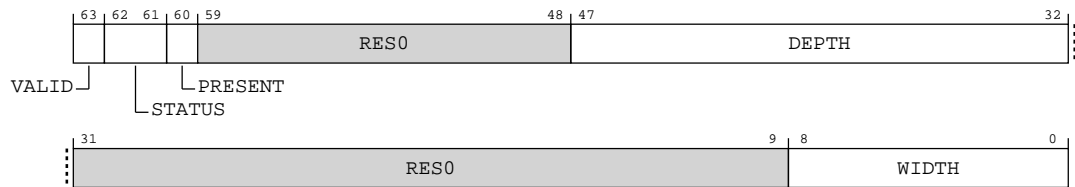


Table 5-401: MBX_DB_ERRINS read bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Indicates if error insertion operation is ongoing 0b0 No error insertion operation is in progress. 0b1 Error insertion operation is in progress.	0b0
[62:61]	STATUS	Result of error insertion when VALID is read as 1'b0 0b00 Error insertion was successful. 0b01 Error insertion encountered out of range address or bit location. 0b10 Error insertion overlap with real ECC error. 0b11 Encoder/decoder mismatch during error insertion.	xx
[60]	PRESENT	If set, indicates that RAM is present and allows inserting errors	x
[59:48]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[47:32]	DEPTH	Defines the maximum RAM address for insertion	16 {x}
[31:9]	RES0	Reserved	RES0
[8:0]	WIDTH	Defines the maximum RAM bit location for insertion	9 {x}

Accessibility

Table 5-402: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0xF010	MBX_DB_ERRINS	None

5.5.2.50 MBX_FST_ERRINS, Mailbox fast channel RAM ECC Error Insertion Register

Enables ECC error insertion in the Mailbox Fast channel RAM. The bit descriptions for this register depend on whether the access is a write or a read.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.MBX

Register offset

0xF018

Bit descriptions

When the access is a write, the MBX_FST_ERRINS register has the following bit assignments.

Figure 5-199: MHUR.MBX_MBX_FST_ERRINS write bit assignments

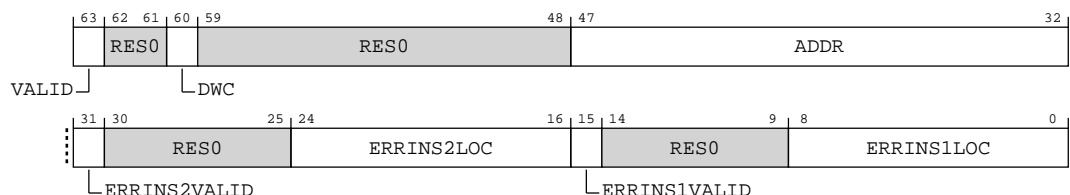


Table 5-403: MBX_FST_ERRINS write bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Writing 1'b1 triggers starting error insertion operation.	x

Bits	Name	Description	Reset
[62:61]	RES0	Reserved	RES0
[60]	DWC	Disable Write Check. If set, tests the ECC encoder.	x
[59:48]	RES0	Reserved	RES0
[47:32]	ADDR	RAM address that required error will be inserted at.	16 {x}
[31]	ERRINS2VALID	If set, enables insertion of second RAM error in bit location specified by ERRINS2LOC	x
[30:25]	RES0	Reserved	RES0
[24:16]	ERRINS2LOC	Defines bit location for second inserted RAM error	9 {x}
[15]	ERRINS1VALID	If set, enables insertion of first RAM error in bit location specified by ERRINS1LOC	x
[14:9]	RES0	Reserved	RES0
[8:0]	ERRINS1LOC	Defines bit location for first inserted RAM error	9 {x}

When the access is a read, the MBX_FST_ERRINS register has the following bit assignments.

Figure 5-200: MHUR.MBX_MBX_FST_ERRINS read bit assignments

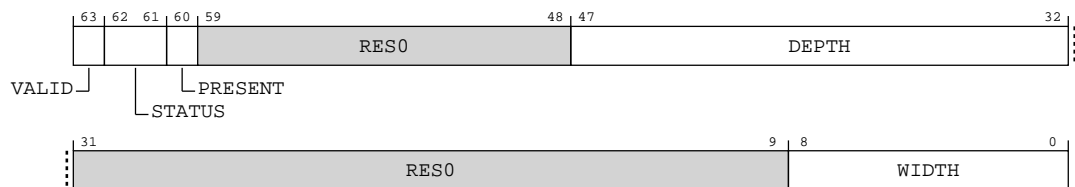


Table 5-404: MBX_FST_ERRINS read bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Indicates if error insertion operation is ongoing 0b0 No error insertion operation is in progress. 0b1 Error insertion operation is in progress.	0b0
[62:61]	STATUS	Result of error insertion when VALID is read as 1'b0 0b00 Error insertion was successful. 0b01 Error insertion encountered out of range address or bit location. 0b10 Error insertion overlap with real ECC error. 0b11 Encoder/decoder mismatch during error insertion.	xx
[60]	PRESENT	If set, indicates that RAM is present and allows inserting errors	x
[59:48]	RES0	Reserved	RES0
[47:32]	DEPTH	Defines the maximum RAM address for insertion	16 {x}
[31:9]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[8:0]	WIDTH	Defines the maximum RAM bit location for insertion	9 {x}

Accessibility

Table 5-405: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0xF018	MBX_FST_ERRINS	None

5.5.2.51 MBX_FCFG_ERRINS, Mailbox FIFO channel config RAM ECC Error Insertion Register

Enables ECC error insertion in the Mailbox FIFO channel configuration RAM. The bit descriptions for this register depend on whether the access is a write or a read.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.MBX

Register offset

0xF020

Bit descriptions

When the access is a write, the MBX_FCFG_ERRINS register has the following bit assignments.

Figure 5-201: MHUR.MBX_MBX_FCFG_ERRINS write bit assignments

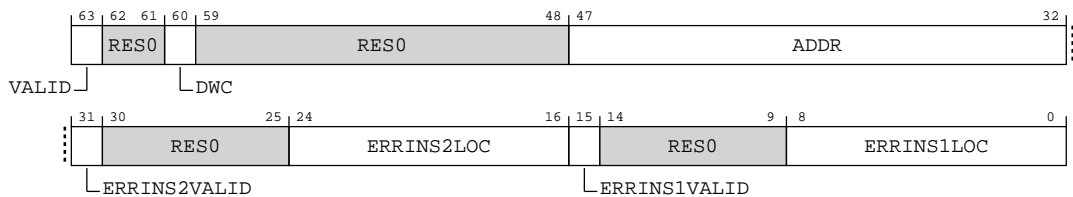


Table 5-406: MBX_FCFG_ERRINS write bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Writing 1'b1 triggers starting error insertion operation.	x
[62:61]	RES0	Reserved	RES0
[60]	DWC	Disable Write Check. If set, tests the ECC encoder.	x

Bits	Name	Description	Reset
[59:48]	RES0	Reserved	RES0
[47:32]	ADDR	RAM address that required error will be inserted at.	16 { x }
[31]	ERRINS2VALID	If set, enables insertion of second RAM error in bit location specified by ERRINS2LOC	x
[30:25]	RES0	Reserved	RES0
[24:16]	ERRINS2LOC	Defines bit location for second inserted RAM error	9 { x }
[15]	ERRINS1VALID	If set, enables insertion of first RAM error in bit location specified by ERRINS1LOC	x
[14:9]	RES0	Reserved	RES0
[8:0]	ERRINS1LOC	Defines bit location for first inserted RAM error	9 { x }

When the access is a read, the MBX_FCFG_ERRINS register has the following bit assignments.

Figure 5-202: MHUR.MBX_MBX_FCFG_ERRINS read bit assignments

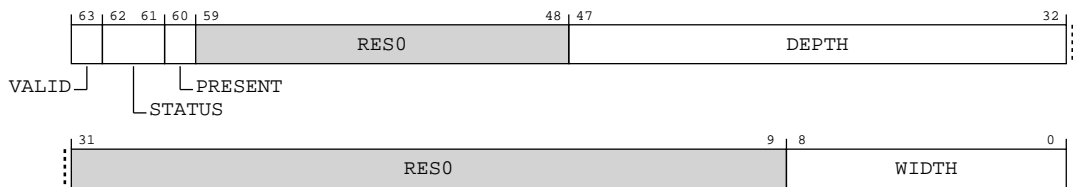


Table 5-407: MBX_FCFG_ERRINS read bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Indicates if error insertion operation is ongoing 0b0 No error insertion operation is in progress. 0b1 Error insertion operation is in progress.	0b0
[62:61]	STATUS	Result of error insertion when VALID is read as 1'b0 0b00 Error insertion was successful. 0b01 Error insertion encountered out of range address or bit location. 0b10 Error insertion overlap with real ECC error. 0b11 Encoder/decoder mismatch during error insertion.	xx
[60]	PRESENT	If set, indicates that RAM is present and allows inserting errors	x
[59:48]	RES0	Reserved	RES0
[47:32]	DEPTH	Defines the maximum RAM address for insertion	16 { x }
[31:9]	RES0	Reserved	RES0
[8:0]	WIDTH	Defines the maximum RAM bit location for insertion	9 { x }

Accessibility

Table 5-408: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0xF020	MBX_FCFG_ERRINS	None

5.5.2.52 MBX_FDATA_ERRINS, Mailbox FIFO channel data RAM ECC Error Insertion Register

Enables ECC error insertion in the Mailbox FIFO channel data RAM. The bit descriptions for this register depend on whether the access is a write or a read.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.MBX

Register offset

0xF028

Bit descriptions

When the access is a write, the MBX_FDATA_ERRINS register has the following bit assignments.

Figure 5-203: MHUR.MBX_MBX_FDATA_ERRINS write bit assignments

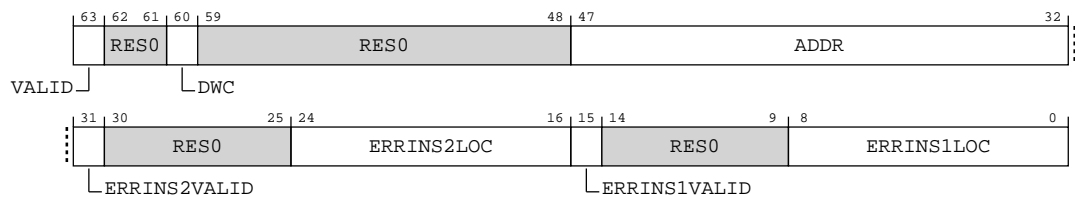


Table 5-409: MBX_FDATA_ERRINS write bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Writing 1'b1 triggers starting error insertion operation.	x
[62:61]	RES0	Reserved	RES0
[60]	DWC	Disable Write Check. If set, tests the ECC encoder.	x
[59:48]	RES0	Reserved	RES0
[47:32]	ADDR	RAM address that required error will be inserted at.	16{x}
[31]	ERRINS2VALID	If set, enables insertion of second RAM error in bit location specified by ERRINS2LOC	x

Bits	Name	Description	Reset
[30:25]	RES0	Reserved	RES0
[24:16]	ERRINS2LOC	Defines bit location for second inserted RAM error	9 {x}
[15]	ERRINS1VALID	If set, enables insertion of first RAM error in bit location specified by ERRINS1LOC	x
[14:9]	RES0	Reserved	RES0
[8:0]	ERRINS1LOC	Defines bit location for first inserted RAM error	9 {x}

When the access is a read, the MBX_FDATA_ERRINS register has the following bit assignments.

Figure 5-204: MHUR.MBX_MBX_FDATA_ERRINS read bit assignments

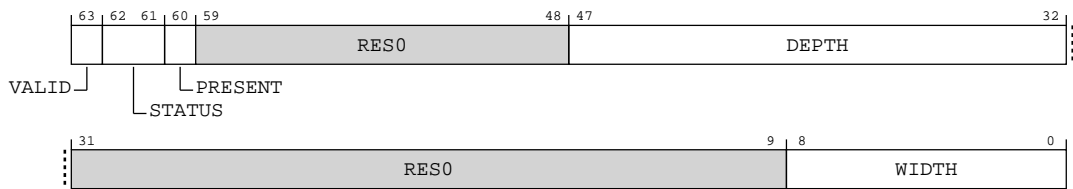


Table 5-410: MBX_FDATA_ERRINS read bit descriptions

Bits	Name	Description	Reset
[63]	VALID	Indicates if error insertion operation is ongoing 0b0 No error insertion operation is in progress. 0b1 Error insertion operation is in progress.	0b0
[62:61]	STATUS	Result of error insertion when VALID is read as 1'b0 0b00 Error insertion was successful. 0b01 Error insertion encountered out of range address or bit location. 0b10 Error insertion overlap with real ECC error. 0b11 Encoder/decoder mismatch during error insertion.	xx
[60]	PRESENT	If set, indicates that RAM is present and allows inserting errors	x
[59:48]	RES0	Reserved	RES0
[47:32]	DEPTH	Defines the maximum RAM address for insertion	16 {x}
[31:9]	RES0	Reserved	RES0
[8:0]	WIDTH	Defines the maximum RAM bit location for insertion	9 {x}

Accessibility

Table 5-411: Accessibility

Component	Offset	Instance	Range
MHUR.MBX	0xF028	MBX_FDATA_ERRINS	None

5.5.3 MHU Receiver RAS register summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MHU Receiver RAS (RRAS) registers.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, see the individual field resets documented on the register description pages or the [Message Handling Unit Architecture version 3.0](#).

Table 5-412: MHUR.RAS register summary

Offset	Name	Type	Reset	Width	Description
(64 * n) + 0x0000	RRAS_ERR<n>FR	RO	See individual bit resets.	64-bit	Error Record <n> Feature Register
(64 * n) + 0x0008	RRAS_ERR<n>CTLR	RW	See individual bit resets.	64-bit	Error Record <n> Control Register
(64 * n) + 0x0010	RRAS_ERR<n>STATUS	RW	See individual bit resets.	64-bit	Error Record <n> Status Register
(64 * n) + 0x0018	RRAS_ERR<n>ADDR	RW	See individual bit resets.	64-bit	Error Record <n> Address Register
(64 * n) + 0x0020	RRAS_ERR<n>MISC0	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 0
(64 * n) + 0x0028	RRAS_ERR<n>MISC1	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 1
(64 * n) + 0x0030	RRAS_ERR<n>MISC2	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 2
(64 * n) + 0x0038	RRAS_ERR<n>MISC3	RW	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 3
0x0E00	RRAS_ERRGSR	RO	See individual bit resets.	64-bit	Error Group Status Register
0x0E10	RRAS_ERRIIDR	RO	See individual bit resets.	32-bit	Implementation Identification Register
0x0E40	RRAS_ERRACR	RW	See individual bit resets.	64-bit	Access Configuration Register
0x0FBC	RRAS_ERRDEVARCH	RO	See individual bit resets.	32-bit	Device Architecture Register
0x0FC8	RRAS_ERRDEVID	RO	See individual bit resets.	32-bit	Device Configuration Register
0x0FD0	RRAS_ERRPIDR4	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 4 Register
0x0FD4	RRAS_ERRPIDR5	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 5 Register
0x0FD8	RRAS_ERRPIDR6	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 6 Register
0x0FDC	RRAS_ERRPIDR7	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 7 Register
0x0FE0	RRAS_ERRPIDR0	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 0 Register
0x0FE4	RRAS_ERRPIDR1	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 1 Register
0x0FE8	RRAS_ERRPIDR2	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 2 Register
0x0FEC	RRAS_ERRPIDR3	RO	See individual bit resets.	32-bit	Receiver RAS Peripheral ID 3 Register
0x0FF0	RRAS_ERRCIDR0	RO	See individual bit resets.	32-bit	Receiver RAS Component ID 0 Register
0x0FF4	RRAS_ERRCIDR1	RO	See individual bit resets.	32-bit	Receiver RAS Component ID 1 Register
0x0FF8	RRAS_ERRCIDR2	RO	See individual bit resets.	32-bit	Receiver RAS Component ID 2 Register

Offset	Name	Type	Reset	Width	Description
0x0FFC	RRAS_ERRCIDR3	RO	See individual bit resets.	32-bit	Receiver RAS Component ID 3 Register

5.5.3.1 RRAS_ERR<n>FR, Error Record <n> Feature Register, n = 0 - 9

Defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

$(64 * n) + 0x0000$

Bit descriptions

Figure 5-205: MHUR_RRAS_ERR<n>FR bit assignments

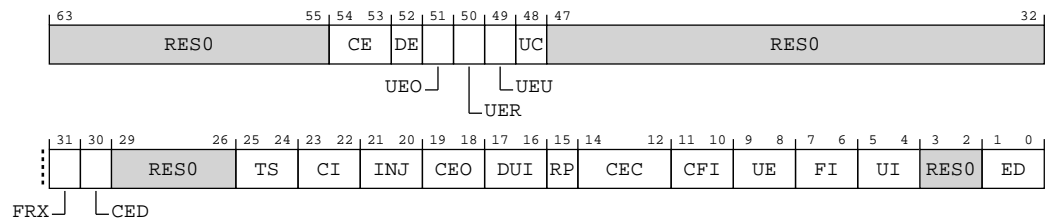


Table 5-413: RRAS_ERR<n>FR bit descriptions

Bits	Name	Description	Reset
[63:55]	RES0	Reserved	RES0
[54:53]	CE	Corrected Error recording. Describes the types of Corrected errors the node can record, if any. 0b00 Does not record Corrected errors. 0b10 Records only non-specific Corrected errors. That is, Corrected errors recorded by setting RRAS_ERR<n>STATUS.CE to 0b10.	xx
[52]	DE	Deferred Error recording. Describes whether the node supports recording Deferred errors. 0b0 Does not record Deferred errors.	0b0

Bits	Name	Description	Reset
[51]	UEO	Latent or Restartable Error recording. Describes whether the node supports recording Latent or Restartable errors. 0b0 Does not record Latent or Restartable errors. 0b1 Records Latent or Restartable errors.	x
[50]	UER	Signaled or Recoverable Error recording. Describes whether the node supports recording Signaled or Recoverable errors. 0b0 Does not record Signaled or Recoverable errors. 0b1 Records Signaled or Recoverable errors.	x
[49]	UEU	Unrecoverable Error recording. Describes whether the node supports recording Unrecoverable errors. 0b0 Does not record Unrecoverable errors.	0b0
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. 0b0 Does not record Uncontainable errors.	0b0
[47:32]	RES0	Reserved	RES0
[31]	FRX	Feature Register extension. Defines whether RRAS_ERR<n>FR[63:48] are architecturally defined. 0b1 RRAS_ERR<n>FR[63:48] are defined by the architecture.	0b1
[30]	CED	Error Counter Disable. Indicates whether the node implements a control to disable any implemented Corrected error counters. 0b0 Enabling and disabling of error counter(s) is not supported. 0b1 Enabling and disabling of error counter(s) is supported and controlled by RRAS_ERR<n>CTLR.CED.	x
[29:26]	RES0	Reserved	RES0
[25:24]	TS	Timestamp Extension. 0b00 Does not support a timestamp register.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node. 0b00 Does not support the critical error interrupt. RRAS_ERR<n>CTLR.CI is RES0 .	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node. 0b00 Does not support the Common Fault Injection Model Extension.	0b00
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by this error record. 0b00 Keeps the previous error syndrome.	0b00

Bits	Name	Description	Reset
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node. 0b00 Does not support the enabling and disabling of error recovery interrupts on deferred errors. RRAS_ERR<n>CTLR.DUI is RES0 .	0b00
[15]	RP	Repeat counter. Indicates whether the node implements a second Corrected error counter in RRAS_ERR<n>MISCO. 0b0 Implements a single Corrected error counter in RRAS_ERR<n>MISCO	0b0
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in RRAS_ERR<n>MISCO. 0b000 Does not implement an corrected error counter. 0b010 Implements an 8-bit Corrected error counter in RRAS_ERR<n>MISCO[39:32].	xxx
[11:10]	CFI	Fault handling interrupt for corrected errors control. Indicates whether the enabling and disabling of fault handling interrupts on corrected errors is supported by the node. 0b00 Does not support the enabling and disabling of fault handling interrupts on corrected errors. RRAS_ERR<n>CTLR.CFI is RES0 .	0b00
[9:8]	UE	In-band error response (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node. 0b00 In-band error response is not supported. 0b10 In-band error response is supported and controllable using RRAS_ERR<n>CTLR.UE.	xx
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node. 0b10 Fault handling interrupt is supported and controllable using RRAS_ERR<n>CTLR.FI.	0b10
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node. 0b00 Does not support the error handling interrupt. RRAS_ERR<n>CTLR.UI is RES0 . 0b10 Error handling interrupt is supported and controllable using RRAS_ERR<n>CTLR.UI.	xx
[3:2]	RES0	Reserved	RES0
[1:0]	ED	Error reporting and logging. Indicates error record <n> is the first record owned the node, and whether the node implements the controls for enabling and disabling error reporting and logging. 0b01 Error reporting and logging always enabled. RRAS_ERR<n>CTLR.ED is RES0 .	0b01

Accessibility

Table 5-414: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	$(64 * n) + 0x0000$	RRAS_ERR<n>FR	None

5.5.3.2 RRAS_ERR<n>CTLR, Error Record <n> Control Register, n = 0 - 9

The error control register contains enable bits for the node that writes to this record.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

$(64 * n) + 0x0008$

Bit descriptions

Figure 5-206: MHUR_RRAS_ERR<n>CTLR bit assignments

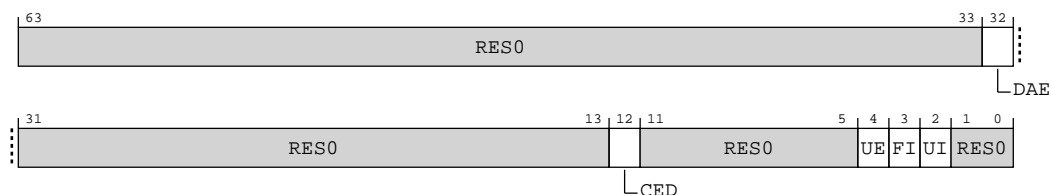


Table 5-415: RRAS_ERR<n>CTLR bit descriptions

Bits	Name	Description	Reset
[63:33]	RES0	Reserved	RES0
[32]	DAE	Disable access errors. Allows disabling reporting of errors due to illegal register accesses. RAZ/WI for all records except Software error record 0. 0b0 Illegal register accesses are treated as errors. 0b1 Reporting of illegal register accesses is disabled.	0b0
[31:13]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[12]	CED	Disable generation of corrected error events from error counters. 0b0 Corrected error events are generated by the error counter. 0b1 Corrected error events are generated when an error is recorded.	0b0
[11:5]	RES0	Reserved	RES0
[4]	UE	In-band error response enable. 0b0 In-band error response for uncorrected errors disabled. 0b1 In-band error response for uncorrected errors enabled.	0b0
[3]	FI	Fault handling interrupt enable. 0b0 Fault handling interrupt disabled. 0b1 Fault handling interrupt enabled.	0b0
[2]	UI	Uncorrected error recovery interrupt enable. 0b0 Error recovery interrupt disabled. 0b1 Error recovery interrupt enabled.	0b0
[1:0]	RES0	Reserved	RES0

Accessibility

Table 5-416: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	(64 * n) + 0x0008	RRAS_ERR<n>CTRL	None

5.5.3.3 RRAS_ERR<n>STATUS, Error Record <n> Status Register, n = 0 - 9

Contains status information for error record <n>.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

$(64 * n) + 0x0010$

Bit descriptions

Figure 5-207: MHUR_RRAS_ERR<n>STATUS bit assignments

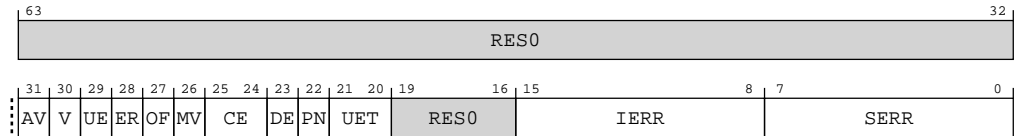


Table 5-417: RRAS_ERR<n>STATUS bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	AV	Address Valid. 0b0 RRAS_ERR<n>ADDR not valid. 0b1 RRAS_ERR<n>ADDR contains an address associated with the highest priority error recorded by this record.	0b0
[30]	V	Status Register Valid. 0b0 RRAS_ERR<n>STATUS not valid. 0b1 RRAS_ERR<n>STATUS valid. At least one error has been recorded.	0b0
[29]	UE	Uncorrected Error. 0b0 No errors have been detected, or all detected errors have been either corrected or deferred. 0b1 At least one detected error was not corrected and not deferred.	0b0
[28]	ER	Error Reported. 0b0 No in-band error response (External Abort) signaled to the Requester making the access. 0b1 An in-band error response was signaled by the component to the Requester making the access.	0b0
[27]	OF	Overflow. Indicates that multiple errors have been detected. 0b0 No error syndrome for an Uncorrected error has been discarded and error counter has not overflowed. 0b1 At least one error syndrome has been discarded or, if an error counter is implemented, it might have overflowed.	0b0

Bits	Name	Description	Reset
[26]	MV	Miscellaneous Registers Valid. 0b0 RRAS_ERR<n>MISC<m> not valid. 0b1 The RRAS_ERR<n>MISC<m> registers contain additional information for an error recorded by this record.	0b0
[25:24]	CE	Corrected Error. 0b00 No errors were corrected. 0b10 At least one error was corrected.	0b00
[23]	DE	Deferred Error. 0b0 No errors were deferred.	0b0
[22]	PN	Poison. 0b0 Error not related to a poison value.	0b0
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. 0b00 Uncorrected error, Uncontainable error (UC). 0b01 Uncorrected error, Unrecoverable error (UEU). 0b10 Uncorrected error, Latent or Restartable error (UEO). 0b11 Uncorrected error, Signaled or Recoverable error (UER).	xx
[19:16]	RES0	Reserved	RES0
[15:8]	IERR	IMPLEMENTATION DEFINED error code. Used with any primary error code RRAS_ERR<n>STATUS.SERR value. See the Reliability, Accessibility and Serviceability chapter for a detailed syndrome description for each MHU error record.	0x00
[7:0]	SERR	Architecturally-defined primary error code. See the RAS System Architecture chapter of the Arm® Architecture Reference Manual for A-profile architecture for more information on this field.	0x00

Accessibility

Table 5-418: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	(64 * n) + 0x0010	RRAS_ERR<n>STATUS	None

5.5.3.4 RRAS_ERR<n>ADDR, Error Record <n> Address Register, n = 0 - 9

If an address is associated with a detected error, then it is written to ERR<n>ADDR when the error is recorded.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

$(64 * n) + 0x0018$

Bit descriptions

Figure 5-208: MHUR_RRAS_ERR<n>ADDR bit assignments

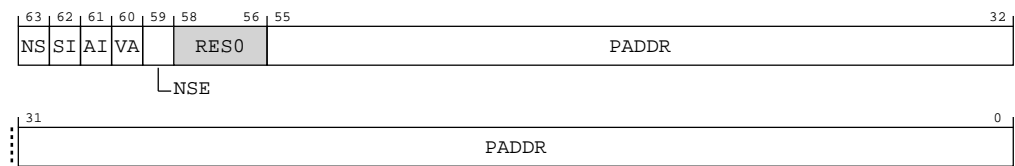


Table 5-419: RRAS_ERR<n>ADDR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. With RRAS_ERR<n>ADDR.NSE, indicates the physical address space of the recorded location. 0b0 When RRAS_ERR<n>ADDR.NSE == 0: RRAS_ERR<n>ADDR.PADDR is a Secure address. When RRAS_ERR<n>ADDR.NSE == 1: RRAS_ERR<n>ADDR.PADDR is a Root address. 0b1 When RRAS_ERR<n>ADDR.NSE == 0: RRAS_ERR<n>ADDR.PADDR is a Non-secure address. When RRAS_ERR<n>ADDR.NSE == 1: RRAS_ERR<n>ADDR.PADDR is a Realm address.	0b0
[62]	SI	Secure Incorrect. Indicates whether RRAS_ERR<n>ADDR.{NS, NSE} are valid. 0b0 RRAS_ERR<n>ADDR.{NS, NSE} are correct. That is, they match the programmers' view of the physical address space for the recorded location.	0b0

Bits	Name	Description	Reset
[61]	AI	Address Incorrect. Indicates whether RRAS_ERR<n>ADDR.PADDR is a valid physical address. 0b0 RRAS_ERR<n>ADDR.PADDR is a valid physical address. That is, it matches the programmers' view of the physical address for the recorded location.	0b0
[60]	VA	Virtual Address. Indicates whether RRAS_ERR<n>ADDR.PADDR field is a virtual address. 0b0 RRAS_ERR<n>ADDR.PADDR is not a virtual address.	0b0
[59]	NSE	Physical Address Space. Together with RRAS_ERR<n>ADDR.NS, indicates the address space for RRAS_ERR<n>ADDR.PADDR.	0b0
[58:56]	RES0	Reserved	RES0
[55:0]	PADDR	Physical Address. Address of the recorded location.	0x0000000000000000

Accessibility

Table 5-420: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	(64 * n) + 0x0018	RRAS_ERR<n>ADDR	None

5.5.3.5 RRAS_ERR<n>MISC0, Error Record <n> Miscellaneous Register 0, n = 0 - 9

Records information on the reported error and error counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

(64 * n) + 0x0020

Bit descriptions

Figure 5-209: MHUR_RRAS_ERR<n>MISC0 bit assignments

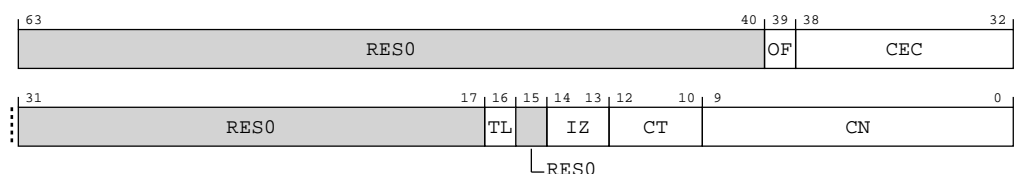


Table 5-421: RRAS_ERR<n>MISC0 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39]	OF	Sticky overflow bit. Set to 1 when RRAS_ERR<n>MISC0.CEC is incremented and wraps through zero. 0b0 Counter has not overflowed. 0b1 Counter has overflowed.	0b0
[38:32]	CEC	Error count. Incremented on for each corrected or uncorrected error.	0b0000000
[31:17]	RES0	Reserved	RES0
[16]	TL	Transfer Lost. 0b0 Error has not caused loss of transfer data in channels within the impact zone. 0b1 Error may have caused loss of transfer data in channels within the impact zone.	0b0
[15]	RES0	Reserved	RES0
[14:13]	IZ	Impact Zone. 0b00 Global. Any channel may be affected by the recorded error. 0b01 Channel specific. Only a specific channel of a given type may be affected by the recorded error as indicated by RRAS_ERR<n>MISC0.CT and RRAS_ERR<n>MISC0.CN. 0b10 Channel type. Any channel of a specific type may be affected by the recorded error as indicated by RRAS_ERR<n>MISC0.CT.	0b00
[12:10]	CT	Channel Type. RES0 when RRAS_ERR<n>MISC0.IZ == 2'b00 0b000 Doorbell channel. 0b001 Fast channel. 0b010 FIFO channel.	0b000
[9:0]	CN	Channel Number. RES0 when RRAS_ERR<n>MISC0.IZ != 2'b01	0b0000000000

Accessibility

Table 5-422: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	(64 * n) + 0x0020	RRAS_ERR<n>MISC0	None

5.5.3.6 RRAS_ERR<n>MISC1, Error Record <n> Miscellaneous Register 1, n = 0 - 9

Records additional information on reported error.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

(64 * n) + 0x0028

Bit descriptions

Figure 5-210: MHUR_RRAS_ERR<n>MISC1 bit assignments

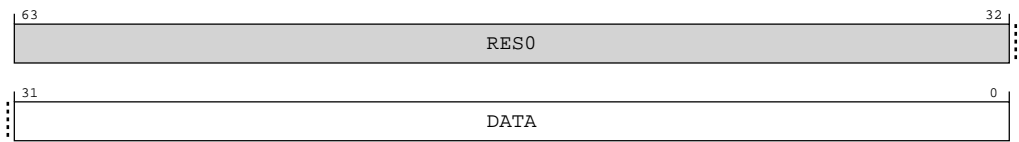


Table 5-423: RRAS_ERR<n>MISC1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	DATA	Additional information on recorded error. See Reliability, Accessibility, and Serviceability , for more information.	0x00000000

Accessibility

Table 5-424: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	(64 * n) + 0x0028	RRAS_ERR<n>MISC1	None

5.5.3.7 RRAS_ERR<n>MISC2, Error Record <n> Miscellaneous Register 2, n = 0 - 9

Records additional information on reported error.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

$(64 * n) + 0x0030$

Bit descriptions

Figure 5-211: MHUR_RRAS_ERR<n>MISC2 bit assignments

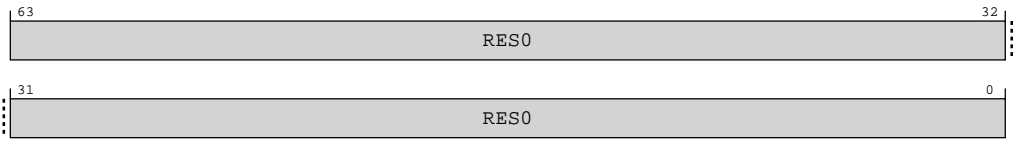


Table 5-425: RRAS_ERR<n>MISC2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	RES0

Accessibility

Table 5-426: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	$(64 * n) + 0x0030$	RRAS_ERR<n>MISC2	None

5.5.3.8 RRAS_ERR<n>MISC3, Error Record <n> Miscellaneous Register 3, n = 0 - 9

Records additional information on reported error.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

$(64 * n) + 0x0038$

Bit descriptions

Figure 5-212: MHUR_RRAS_ERR<n>MISC3 bit assignments

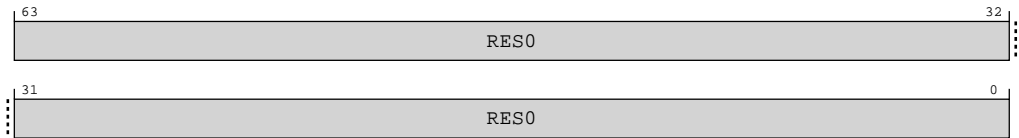


Table 5-427: RRAS_ERR<n>MISC3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	RES0

Accessibility

Table 5-428: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	(64 * n) + 0x0038	RRAS_ERR<n>MISC3	None

5.5.3.9 RRAS_ERRGSR, Error Group Status Register

Shows the status for the records in the group.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

MHUR.RRAS

Register offset

0x0E00

Bit descriptions

Figure 5-213: MHUR_RRAS_ERRGSR bit assignments

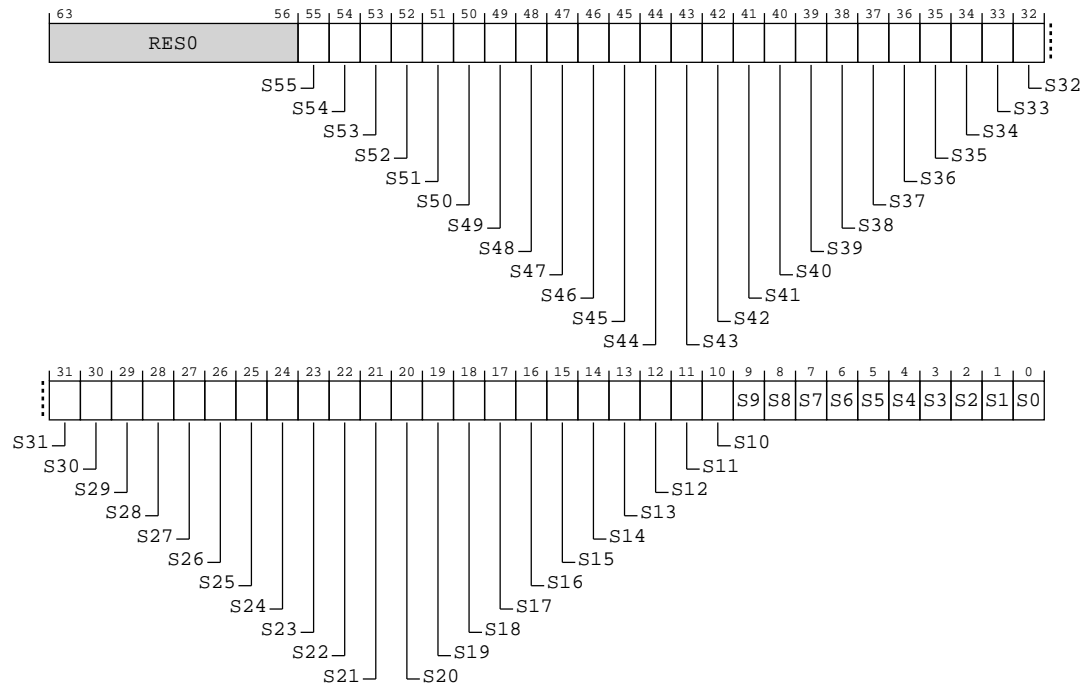


Table 5-429: RRAS_ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:0]	S<m>, bit[m], where m = 55 to 0	<p>The status for error record <m>. A read-only copy of RRAS_ERR<m>STATUS.V.</p> <p>0b0 No error.</p> <p>0b1 One or more errors.</p>	0x0000000000000000

Accessibility

Table 5-430: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0E00	RRAS_ERRGSR	None

5.5.3.10 RRAS_ERRIIDR, Implementation Identification Register

Defines the implementer of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0E10

Bit descriptions

Figure 5-214: MHUR_RRAS_ERRIIDR bit assignments

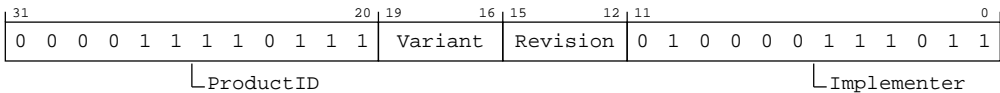


Table 5-431: RRAS_ERRIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component. 0b000011110111	0x0F7
[19:16]	Variant	Component major revision. This field distinguishes product variants or major revisions of the product.	The reset value depends on the product version used. <ul style="list-style-type: none">0x0 - r0
[15:12]	Revision	Component minor revision. This field distinguishes minor revisions of the product.	The reset value depends on the product version used. <ul style="list-style-type: none">0x0 - p00x1 - p1
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the RAS component. For an Arm implementation, this field has the value 0x43B. 0b010000111011	0x43B

Accessibility

Table 5-432: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0E10	RRAS_ERRIIDR	None

5.5.3.11 RRAS_ERRACR, Access Configuration Register

Controls visibility of error records.

Configurations

This register is present only when TZE is implemented for the MHU Receiver. Otherwise, direct accesses to RRAS_ERRACR are RAZ/WI.

Attributes

Width
64

Component
MHUR.RRAS

Register offset
0x0E40

Bit descriptions

Figure 5-215: MHUR_RRAS_ERRACR bit assignments

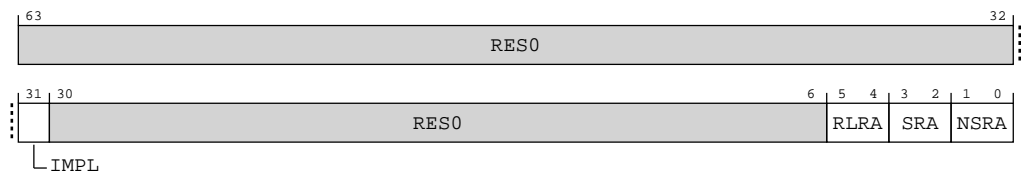


Table 5-433: RRAS_ERRACR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	IMPL	Indicates RRAS_ERRACR is present. 0b1 RRAS_ERRACR is present.	0b1
[30:6]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[5:4]	RLRA	When RME is implemented for the MHU Receiver Realm Restricted Access. Controls Realm access to error records when RME is supported. 0b00 Realm access is disabled 0b01 Realm read access is enabled. Realm writes are ignored. 0b11 Realm read/write access is allowed. Otherwise RESO	xx
[3:2]	SRA	Secure Restricted Access. Controls Secure access to error records when RME is supported. 0b00 Secure access is disabled 0b01 Secure read access is enabled. Realm writes are ignored. 0b11 Secure read/write access is allowed.	0b11
[1:0]	NSRA	Non-secure Restricted Access. Controls Non-secure access to error records when TZE is supported. 0b00 Non-secure access is disabled 0b01 Non-secure read access is enabled. Realm writes are ignored. 0b11 Non-secure read/write access is allowed.	0b11

Accessibility

Table 5-434: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0E40	RRAS_ERRACR	None

5.5.3.12 RRAS_ERRDEVARCH, Device Architecture Register

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FBC

Bit descriptions

Figure 5-216: MHUR_RRAS_ERRDEVARCH bit assignments

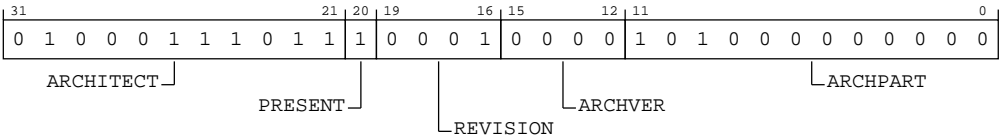


Table 5-435: RRAS_ERRDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. 0b01000111011 JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	0b01000111011
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. 0b1 Device Architecture information present.	0b1
[19:16]	REVISION	Revision. Defines the architecture revision of the component. 0b0001 RAS System Architecture v1.1.	0b0001
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. 0b0000 RAS System Architecture v1.	0b0000
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. 0b101000000000 RAS System Architecture.	0xA00

Accessibility

Table 5-436: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FBC	RRAS_ERRDEVARCH	None

5.5.3.13 RRAS_ERRDEVID, Device Configuration Register

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FC8

Bit descriptions

Figure 5-217: MHUR_RRAS_ERRDEVID bit assignments



Table 5-437: RRAS_ERRDEVID bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:0]	NUM	Highest numbered index of the error records in this group, plus one.	0x4

Accessibility

Table 5-438: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FC8	RRAS_ERRDEVID	None

5.5.3.14 RRAS_ERRPIDR4, Receiver RAS Peripheral ID 4 Register

Returns byte[4] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
MHUR.RRAS

Register offset
0x0FD0

Bit descriptions

Figure 5-218: MHUR_RRAS_ERRPIDR4 bit assignments



Table 5-439: RRAS_ERRPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	Size of the component 0b0000 The size of the component must be identified using a combination of the following registers: <ul style="list-style-type: none">Peripheral ID 0-7 registersComponent ID 0-3 registers<x>_FEAT_SPT0 and <x>_FEAT_SPT1 registers	0b0000
[3:0]	DES_2	JEP106 continuation code 0b0100	0b0100

Accessibility

Table 5-440: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FD0	RRAS_ERRPIDR4	None

5.5.3.15 RRAS_ERRPIDR5, Receiver RAS Peripheral ID 5 Register

Returns byte[5] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUR.RRAS

Register offset
0x0FD4

Bit descriptions

Figure 5-219: MHUR_RRAS_ERRPIDR5 bit assignments

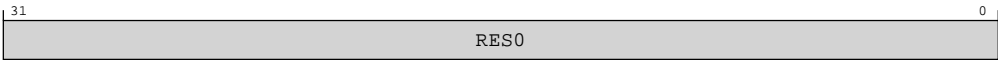


Table 5-441: RRAS_ERRPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-442: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FD4	RRAS_ERRPIDR5	None

5.5.3.16 RRAS_ERRPIDR6, Receiver RAS Peripheral ID 6 Register

Returns byte[6] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUR.RRAS

Register offset
0x0FD8

Bit descriptions

Figure 5-220: MHUR_RRAS_ERRPIDR6 bit assignments

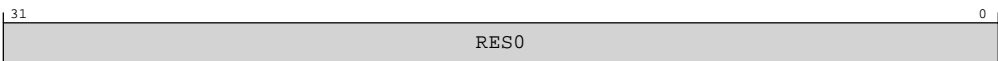


Table 5-443: RRAS_ERRPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-444: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FD8	RRAS_ERRPIDR6	None

5.5.3.17 RRAS_ERRPIDR7, Receiver RAS Peripheral ID 7 Register

Returns byte[7] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FDC

Bit descriptions

Figure 5-221: MHUR_RRAS_ERRPIDR7 bit assignments



Table 5-445: RRAS_ERRPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Table 5-446: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FDC	RRAS_ERRPIDR7	None

5.5.3.18 RRAS_ERRPIDR0, Receiver RAS Peripheral ID 0 Register

Returns byte[0] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FE0

Bit descriptions

Figure 5-222: MHUR_RRAS_ERRPIDR0 bit assignments

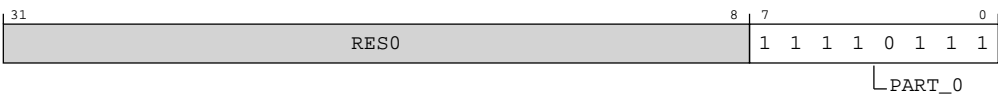


Table 5-447: RRAS_ERRPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Bits 7:0 of the Part ID for the implementation of the MHU 0b11110111	0xF7

Accessibility

Table 5-448: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FE0	RRAS_ERRPIDR0	None

5.5.3.19 RRAS_ERRPIDR1, Receiver RAS Peripheral ID 1 Register

Returns byte[1] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUR.RRAS

Register offset
0x0FE4

Bit descriptions

Figure 5-223: MHUR_RRAS_ERRPIDR1 bit assignments



Table 5-449: RRAS_ERRPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	DES_0	Bits 3:0 of the JEP106 identification code 0b1011	0b1011
[3:0]	PART_1	Bits 11:8 of the Part ID for the implementation of the MHU 0b0000	0b0000

Accessibility

Table 5-450: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FE4	RRAS_ERRPIDR1	None

5.5.3.20 RRAS_ERRPIDR2, Receiver RAS Peripheral ID 2 Register

Returns byte[2] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
MHUR.RRAS

Register offset
0x0FE8

Bit descriptions

Figure 5-224: MHUR_RRAS_ERRPIDR2 bit assignments

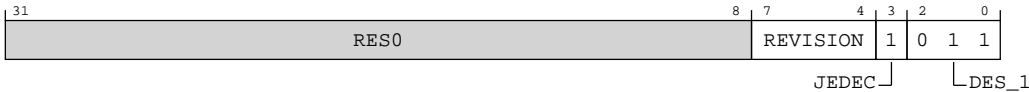


Table 5-451: RRAS_ERRPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	The REVISION field is an incremental value starting at 0x0 for the first design of a component. The value is increased by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision.	The reset value depends on the product version used. <ul style="list-style-type: none">0x0 - r0p00x1 - r0p1
[3]	JEDEC	Must be 0b1 to indicate that a JEDEC-assigned value is used. 0b1	0b1
[2:0]	DES_1	JEP106 identification bits [6:4] 0b011	0b011

Accessibility

Table 5-452: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FE8	RRAS_ERRPIDR2	None

5.5.3.21 RRAS_ERRPIDR3, Receiver RAS Peripheral ID 3 Register

Returns byte[3] of the peripheral ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FEC

Bit descriptions

Figure 5-225: MHUR_RRAS_ERRPIDR3 bit assignments



Table 5-453: RRAS_ERRPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	The REVAND field indicates minor errata fixes specific to this design, for example metal fixes after implementation. Usually this field is zero. Together with PIDR2.REVISION, PIDR3.REVAND forms the revision number of the component.	0x0
[3:0]	CMOD	Customer Modified. If the component is reusable IP, the CMOD field indicates whether the customer has modified the behavior of the component. Arm recommends that the user or debugger reads the documentation for the component to determine the modifications that are made to the component. For any two components with the same values of the Peripheral ID 0-7 and Component ID 0-3 registers <ul style="list-style-type: none"> If the value of the CMOD fields of both components equals zero, the components are identical If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have been subjected to the same modifications. If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same values of the Peripheral ID 0-7 and Component ID 0-3 registers 	0x0

Accessibility

Table 5-454: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FEC	RRAS_ERRPIDR3	None

5.5.3.22 RRAS_ERRCIDR0, Receiver RAS Component ID 0 Register

Returns byte[0] of the component ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FF0

Bit descriptions

Figure 5-226: MHUR_RRAS_ERRCIDR0 bit assignments

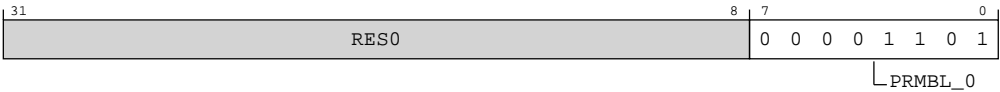


Table 5-455: RRAS_ERRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble, segment 0 0b00001101	0x0D

Accessibility

Table 5-456: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FF0	RRAS_ERRCIDR0	None

5.5.3.23 RRAS_ERRCIDR1, Receiver RAS Component ID 1 Register

Returns byte[1] of the component ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FF4

Bit descriptions

Figure 5-227: MHUR_RRAS_ERRCIDR1 bit assignments

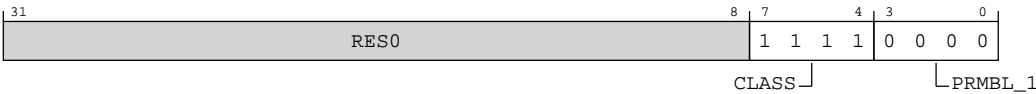


Table 5-457: RRAS_ERRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component Class 0b1111 CoreLink, PrimeCell, or system component with no standardized register layout, for backwards compatibility.	0b1111
[3:0]	PRMBL_1	Preamble, segment 1 0b0000	0b0000

Accessibility

Table 5-458: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FF4	RRAS_ERRCIDR1	None

5.5.3.24 RRAS_ERRCIDR2, Receiver RAS Component ID 2 Register

Returns byte[2] of the component ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FF8

Bit descriptions

Figure 5-228: MHUR_RRAS_ERRCIDR2 bit assignments

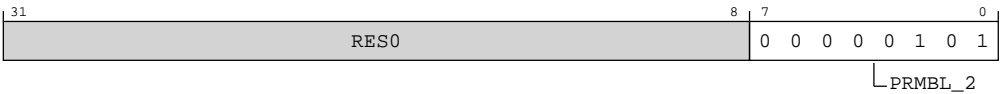


Table 5-459: RRAS_ERRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble, segment 2 0b00000101	0x05

Accessibility

Table 5-460: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FF8	RRAS_ERRCIDR2	None

5.5.3.25 RRAS_ERRCIDR3, Receiver RAS Component ID 3 Register

Returns byte[3] of the component ID for Receiver RAS page.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

MHUR.RRAS

Register offset

0x0FFC

Bit descriptions

Figure 5-229: MHUR_RRAS_ERRCIDR3 bit assignments

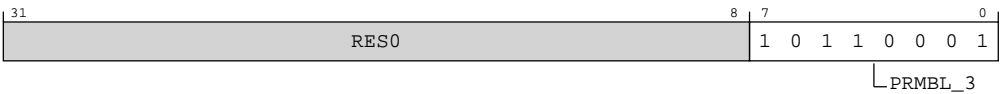


Table 5-461: RRAS_ERRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble, segment 3 0b10110001	0xB1

Accessibility

Table 5-462: Accessibility

Component	Offset	Instance	Range
MHUR.RRAS	0x0FFC	RRAS_ERRCIDR3	None

6. Functional safety features of MHU-320AE

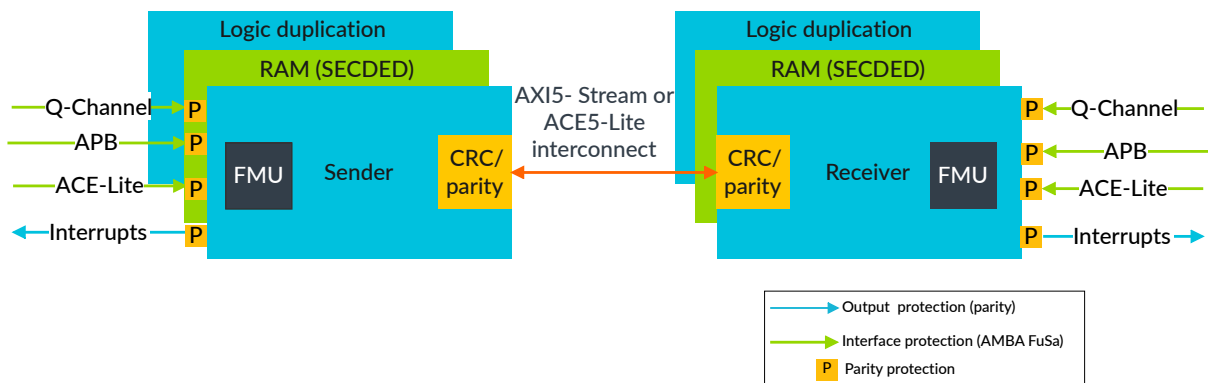
MHU-320AE functional safety features are present in MHU-320AE configurations, where `FUSA_PRESENT == 1`.

6.1 Protection mechanisms of MHU-320AE

The MHU-320AE provides built-in protection mechanisms.

The following figure shows location of the MHU-320AE main protection mechanisms.

Figure 6-1: Protection mechanism distribution



MHU-320AE contains the following FuSa protection mechanisms.

Lock-step logic protection

The logic is protected with duplicated logic running in lock-step.

RAM protection

The RAMs are shared between the duplicated blocks and are protected with SECDED ECC. The address is further protected with parity.

AMBA AXI5-Stream interconnect protection

The AXI5-Stream interconnect that connects the MHU-320AE blocks, is protected with either:

- AMBA parity for simple point-to-point connections
- End-to-end CRC for packets routed over other interconnects or when ADB domain bridges or register slices are required

AMBA external interface protection

All external AMBA interfaces are protected with AMBA parity signals. AMBA parity protects point-to-point connections consisting of wires and buffers only, and no gates. This protection includes the ACE5-Lite, AXI5-Stream, Q-Channel, and APB external ports.

Q-Channel protection

The Q-Channel is protected by AMBA parity.



Figure 6-1: Protection mechanism distribution on page 330 shows Q-Channel parity protection that is enabled on only one ITS block. However, when `QCH_PROTECTION_TYPE == 1`, the Q-Channel protection is present for all Q-Channel interfaces.

Systematic fault watchdog

MHU-320AE contains a watchdog-based PING/ACK mechanism. This mechanism protects against systematic errors on the interconnect that connects the various blocks. If the mechanism does not receive a response within the programmable timeout window, it reports a fault.

The PING/ACK mechanism is part of CRC end-to-end protection, where separate CRC packets are sent to check the data integrity of data packets and protect against spurious packets and packet loss.

Clocks and resets

The clocks and resets are duplicated. The internally gated clocks operate with a temporal delay of two. That is, the secondary logic operates two cycles later than the primary logic.

Fault Management Unit

The Fault Management Unit (FMU) resides in the MHU Sender, in the MHU Receiver, or in both, depending on the MHU configuration. It processes faults that the protection mechanisms detect from all MHU-320AE blocks. The FMU records the fault syndrome in the error records and reports the fault using Error Recovery Interrupt (ERI) and Critical Error Interrupt (CRI). There are also FMU registers that enable fault injection and clearing for each protection mechanism. The FMU communicates with an external Safety Island through an APB port. The APB port is for FuSa purposes.

6.2 Fault management unit

The Fault Management Unit (FMU) implements the following functionality in MHU-320AE.

The FMU implements the following functionality in MHU-320AE:

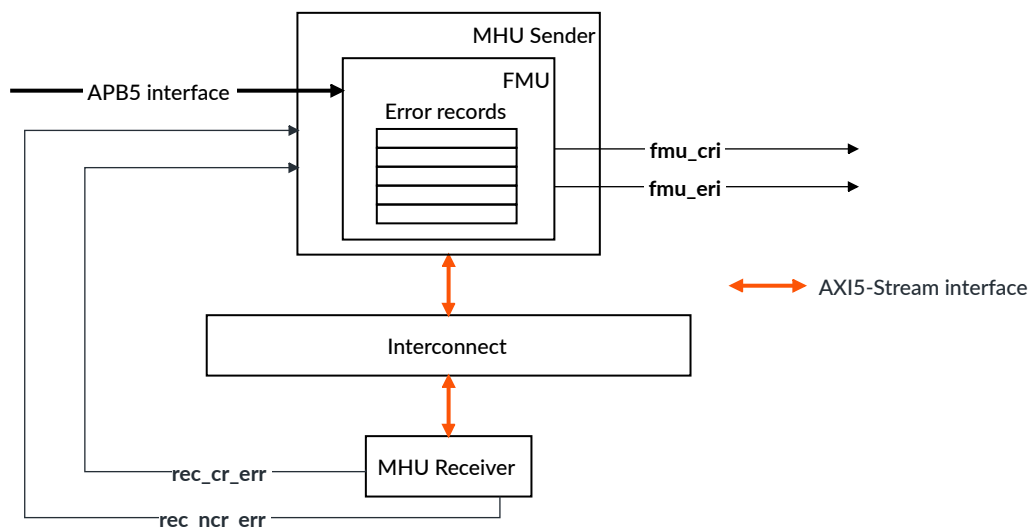
- Dedicated APB5 interface to access error records and other registers.
- Routes all errors to the Safety Island, if enabled.

- Provides software the means to enable or disable a protection mechanism within a MHU-320AE block.
- Receives error signaling from all protection mechanisms within other MHU-320AE blocks.
- Maintains error records for each MHU-320AE block type, for software inspection and provides information on the source of the error.
- Retains error records across functional reset.
- Enables software error recovery testing by providing error injection capabilities in a protection mechanism.

The `FMU_LOCATION` configuration parameter determines whether the FMU is a part of the MHU Sender or MHU Receiver hierarchy or both.

The following figure shows the FMU and its interconnections when the FMU is located in the MHU Sender.

Figure 6-2: FMU



6.2.1 FMU APB5 interface

The programmer view registers inside the FMU are accessible through an APB5 interface that is protected with AMBA parity extensions.

The APB5 completer interface width is 32 bits. Some of the FMU registers are 64 bits wide, so two 32-bit APB accesses, in any order, are necessary for reads or writes of those registers.

The APB5 port allows only Secure access to the FMU. To implement this access restriction, the `pprot[1]` signal is checked during an access. If the access fails the security check, the MHU-320AE does not use the `pslverr` signal to indicate this error condition, the `pslverr` signal remains LOW.

6.2.2 Error signaling from a MHU-320AE block to the FMU

MHU-320AE implements several protection mechanisms in each MHU block to protect against random transient or permanent errors. Each protection mechanism sends an error signal to the fault collator in its MHU block. The MHU block then forwards the error signal to the FMU in the central MHU Distributor using the existing AXI5-Stream interface.

In addition to reporting errors through the AXI5-Stream interconnect, each remote MHU block has a `cr_err_out` and `ncr_err_out` output signal that indicates either a critical or non-critical error within its block.

As the MHU exits reset, it sets all protection mechanisms to report a critical error, except for the RAM SEC protection mechanisms that report a non-critical error. Software can use `FMU_SMCR` register details to alter these criticality assignments.

Connect the `*_err_out` and `*_ncr_err_out` signals from the MHU component that does not contain the FMU to the corresponding signals in the MHU component that contains the FMU.

6.2.3 Error signaling by the FMU

When a protection mechanism detects an error, it forwards the error to the FMU. If the FMU is enabled, it signals the error to the entire system using the error interrupt signals.

The FMU uses the following error interrupt signals:

- Error recovery interrupt, `fmu_eri` signal (ERI)
- Critical error interrupt, `fmu_cri` signal (CRI)

The ERI and CRI interrupts are enabled on reset. Software can use `FMU_ERR<n>CTLR` to disable the error reporting through ERI or CRI.

Non-critical errors are reported using ERI and critical errors are reported using CRI. The `FMU_ERR<n>CTLR.CI` and `FMU_ERR<n>CTLR.UI` bits control this reporting.

Critical errors and non-critical errors can be redirected to different error recovery handlers.

6.2.4 Error record format

The FMU contains one error record for each MHU-320AE block type.

MHU-320AE faults are recorded in error records.

The error record registers are accessible through the APB5 interface on the corresponding FMU. The FMU has a different reset input signal than the MHU, so that the error record retains its state even when the MHU block is being reset.

The following table lists the error record block type IDs. The same record numbering is maintained irrespective of FMU location.

Table 6-1: MHU-320AE error record block type IDs

FMU_ERRUPDATE.RECORD_PAIR	Block type	Error record ID, <n>	Error criticality
0	MHU Sender	0	Critical
0	MHU Sender	1	Non-critical
1	MHU Receiver	2	Critical
1	MHU Receiver	3	Non-critical
2	FMU	4	Critical
2	FMU	5	Non-critical

6.2.5 FMU reset

When the FMU reports multiple uncorrectable errors, the error recovery procedure might require the MHU to be reset. To facilitate this situation, the FMU has a separate reset input signal, `fmu_reset_n`.

This reset differs from the MHU functional reset, `reset_n` signal. It allows the FMU to retain error records across MHU functional reset.

The FMU is reset together with the block where it is placed.

When the FMU and its respective block are powered down, the system software must not allow outstanding APB5 accesses to reach the FMU.

6.2.6 Protection mechanism IDs

The MHU-320AE assigns an ID for each protection mechanism in the MHU Sender and MHU Receiver. For each protection mechanism ID, we provide a description and the recommended recovery process.

MHU Sender protection mechanisms

The following table lists the MHU Sender protection IDs.

Table 6-2: MHU Sender protection IDs

ID	Protection name	Description	Recovery
0	INVALID	Reserved	-
1	SM_CLOCK_SND	Clock error	Block reset
2	SM_RESET_SND	Reset error	Reset error recovery
3	SM_LOCKSTEP_SND	Lockstep error	Block reset
4	SM_QCH_CLK_SND	Clock Q-Channel error	Q-Channel error recovery
5	SM_QCH_PWR_SND	Power Q-Channel error	Q-Channel error recovery

ID	Protection name	Description	Recovery
6	SM_REGPAR_SND	Register interface parity error	Block reset
7	SM_AXITPAR_SND	MHU-Stream port parity error	Block reset
8	SM_AXITCRC_SND	MHU-Stream port CRC error	CRC error recovery
9	SM_ACELSPAR_SND	ACE-Lite MHU-Stream subordinate port parity error	Block reset
10	SM_ACELMPAR_SND	ACE-Lite MHU-Stream manager port parity error	Block reset
11	SM_LPD_CLK_SND	Clock Q-Channel LPD error	LPD error recovery
12	SM_LPD_CLK_UNP_SND	Clock Q-Channel unprotected LPD error	LPD error recovery
13	SM_LPD_PWR_SND	Power Q-Channel LPD error	LPD error recovery
14	SM_LPD_PWR_UNP_SND	Power Q-Channel unprotected LPD error	LPD error recovery
15	SM_DFT_SND	DFT interface error	Block reset
16	SM_MBIST_SND	MBIST interface error	Block reset
17	SM_SECD_SND_FIFO	FIFO channel RAM SEC in data bit	RAM error recovery
18	SM_SECA_SND_FIFO	FIFO channel RAM SEC in address bit	RAM error recovery
19	SM_DED_SND_FIFO	FIFO channel RAM DED	RAM error recovery
20	SM_EXT0_SND	External error 0	External error recovery
21	SM_EXT1_SND	External error 1	External error recovery

MHU Receiver protection mechanisms

The following table lists the MHU Receiver protection IDs.

Table 6-3: MHU Receiver protection IDs

ID	Protection name	Description	Recovery
0	INVALID	Reserved	-
1	SM_CLOCK_REC	Clock error	Block reset
2	SM_RESET_REC	Reset error	Reset error recovery
3	SM_LOCKSTEP_REC	Lockstep error	Block reset
4	SM_QCH_CLK_REC	Clock Q-Channel error	Q-Channel error recovery
5	SM_QCH_PWR_REC	Power Q-Channel error	Q-Channel error recovery
6	SM_REGPAR_REC	Register interface parity error	Block reset
7	SM_AXITPAR_REC	MHU-Stream port parity error	Block reset
8	SM_AXITCRC_REC	MHU-Stream port CRC error	CRC error recovery
9	SM_ACELSPAR_REC	ACE-Lite MHU-Stream subordinate port parity error	Block reset
10	SM_ACELMPAR_REC	ACE-Lite MHU-Stream manager port parity error	Block reset
11	SM_LPD_CLK_REC	Clock Q-Channel LPD error	LPD error recovery
12	SM_LPD_CLK_UNP_REC	Clock Q-Channel unprotected LPD error	LPD error recovery
13	SM_LPD_PWR_REC	Power Q-Channel LPD error	LPD error recovery
14	SM_LPD_PWR_UNP_REC	Power Q-Channel unprotected LPD error	LPD error recovery
15	SM_DFT_REC	DFT interface error	Block reset
16	SM_MBIST_REC	MBIST interface error	Block reset
17	SM_SECD_REC_DB	Doorbell channel RAM SEC in data bit	RAM error recovery
18	SM_SECA_REC_DB	Doorbell channel RAM SEC in address bit	RAM error recovery

ID	Protection name	Description	Recovery
19	SM_DED_REC_DB	Doorbell channel RAM DED	RAM error recovery
20	SM_SECD_REC_FAST	Fast channel RAM SEC in data bit	RAM error recovery
21	SM_SECA_REC_FAST	Fast channel RAM SEC in address bit	RAM error recovery
22	SM_DED_REC_FAST	Fast channel RAM DED	RAM error recovery
23	SM_SECD_REC_CFIFO	FIFO channel config RAM SEC in data bit	RAM error recovery -
24	SM_SECA_REC_CFIFO	FIFO channel config RAM SEC in address bit	RAM error recovery
25	SM_DED_REC_CFIFO	FIFO channel config RAM DED	RAM error recovery
26	SM_SECD_REC_DFIFO	FIFO channel data RAM SEC in data bit	RAM error recovery
27	SM_SECA_REC_DFIFO	FIFO channel data RAM SEC in address bit	RAM error recovery
28	SM_DED_REC_DFIFO	FIFO channel data RAM DED	RAM error recovery
29	SM_EXT0_REC	External error 0	External error recovery
30	SM_EXT1_REC	External error 1	External error recovery

FMU protection mechanisms

The following table lists the IDs for each protection mechanism of the FMU.

Table 6-4: FMU protection mechanisms

ID	Protection name	Description	Recovery
0	INVALID	Reserved	-
1	SM_CLOCK_FMU	Clock error	Block reset
2	SM_RESET_FMU	Reset error	Reset error recovery
3	SM_LOCKSTEP_FMU	Lockstep error	Block reset
4	SM_QCH_FMU	Q-Channel error	Q-Channel error recovery
5	SM_APBPTY_FMU	AMBA parity error	FMU APB recovery
6	SM_DFT_FMU	DFT protection error	FMU reset
7	SM_BRIDGEFMU_FMU	A consistency error on the FMU-side of the MHU-FMU bridge.	Full reset
8	SM_KEY_FMU	MHU-Stream port CRC error	CRC error recovery
9	SM_SECURITY_FMU	Non-secure read or write access to a Secure FMU register.	FMU APB access error recovery
10	SM_APB_ACCESS_FMU	APB write error to an FMU register. This error occurs when any of the following is true: <ul style="list-style-type: none"> A write to an invalid address. An invalid address is defined as a register that is not shown in the MHU FMU register block register summary A write to a read-only FMU register. Read accesses cannot generate this error. Therefore, for discovery purposes, software can always read any FMU RAS registers. 	FMU APB access error recovery

ID	Protection name	Description	Recovery
11	SM_APB_FIELD_FMU	APB field invalid. This error occurs when any of the following is true: <ul style="list-style-type: none"> BLKTYPE field value selects a block type that the MHU configuration does not support. BLKTYPE > 2 PAGEID field value > 15 BLKID != 0 PROTID == 0. The MHU uses this field setting to indicate that an AXI5-Stream packet is invalid. 	FMU APB access error recovery
12	SM_APB_SIZE_FMU	APB size invalid. This error occurs for any FMU sparse write access, that is, the pstrb signal is not set to 0b1111. The MHU ignores any sparse write accesses to the FMU registers.	FMU APB access error recovery
13	SM_BUSY_FMU	APB access discarded due to FMU busy error. This error occurs when FMU_STATUS.BUSY==1 and software accesses an FMU register that requires the FMU to set BUSY to 1.	FMU APB access error recovery
14	SM_BRIDGEMHU_FMU	A consistency error on the MHU-side of the MHU-FMU bridge. Asynchronous REQ/ACK error, so PROTID must be higher.	FMU reset
255	-	Software can use this setting to: <ul style="list-style-type: none"> Enable or disable the error output signals on the FMU. See Enable or disable both error signals on a block. Resend any outstanding errors on the FMU. See Discover the active errors on the block. 	-

6.3 Error recovery procedures

When a protection mechanism is triggered then it might be necessary for software to perform a recovery procedure, so that the MHU-320AE or system can continue functioning. The following sections provide guidance about the recovery process that we recommend for the MHU Sender and MHU Receiver protection mechanism IDs.

Asynchronous input error recovery

The MHU does not need to be reset after an asynchronous input error because corrupt input transitions are contained by the input protection logic. The incoming event might not have been sampled, but architectural MHU operation is unaffected.

Block reset

This reset applies to all blocks except the corresponding FMU.

The block that reported the error must be reset and also the corresponding FMU. Follow the steps in the [Power management](#) section to get all the blocks into the defined state before resetting them.

It is possible that the system is in a state where it is unable to complete the full powerdown sequence and a complete system reset is required.

CRC error recovery

Read the error type from the AXI5-Stream protection block that is reporting the error. If the error type is a:

CRC timeout error

Increase the CRC timeout and continue. If the timeout is repeatable, then it might require a block reset.

CRC error

Perform a block reset, as described in Block reset.

External error recovery

An external error does not indicate an issue within the MHU. The system integrator is responsible for ensuring that recovery is sufficient for the external error source, which might include resetting the MHU or the entire system.

FMU APB access error recovery

If the FMU has detected an incorrect access type, then there is no specific recovery procedure needed. The error has been contained by the FMU. If software was attempting an update when this error occurred, then software should repeat the update.

FMU APB recovery

If an APB parity error occurred on the register write, then perform the write again. Data in the FMU might have been corrupted.

FMU reset

Reset the FMU. Ideally the FMU Q-Channel must be quiesced before reset is applied. If resetting while not in the QSTOPPED state, then it violates the Q-Channel protocol, so an unexpected response might be logged in the FMU when exiting reset.

Full reset

Software initiates the quiesce procedure and then resets the MHU-320AE. If the MHU-320AE fails to respond to the powerdown sequence, then a full system reset is required.

LPD error recovery

The LPD recovery sequence depends on whether the error type as follows:

LPD error PROTID

This error requires a full reset.

LPD MHU interconnect error PROTID

This error requires a MHU reset.

Unprotected devices are AXI5-Stream interconnect components within the MHU top-level and a fault here is contained within the MHU.

Q-Channel error recovery

The MHU-320AE does not need to be reset after a Q-Channel error because corrupt transitions on the qreqn signal are contained by the Q-Channel protection logic. In the event of a qreqn signal fault, the MHU-320AE does not change its state. If in QSTOPPED, the MHU-320AE remains there and a full reset is necessary.

If the Q-Channel error is persistent, steps should be taken to ensure that the clock that the Q-Channel controls continues to run, if necessary, by keeping it running at all times.

RAM error recovery

For SECD RAM errors, no recovery procedure is required. For SECA and DED RAM errors, a recovery procedure is required. The recovery procedure is different for each type of RAM, see the following procedures:

- [Doorbell channel error recovery procedure](#)
- [Fast channel error recovery procedure](#)
- [FIFO channel error recovery procedure](#)

Some RAM errors are lossy and the system integrator must determine the system recovery behavior for each RAM type. RAM error reports are also duplicated in the corresponding SRAS or RRAS address space.

Reset error recovery

Reset errors are contained within reset protection, so no recovery procedure is required.

Wire-only error reported recovery

When reading FMU_ERR<n>STATUS, if V=1 and W=1 but PROTID=0 (IERR=0) is observed, software should continue to read FMU_ERR<n>STATUS until either PROTID is not 0 or a timeout is reached. The timeout should be determined by the maximum AXI5-Stream fabric packet delay. If a timeout occurs, then either there is a fault with being able to receive packets from MHU-320AE blocks or there is a fault with the error wire and so the W bit reporting.

To determine if packets can be received properly, access FMU_ERRUPDATE to cause all errors to be resent for that error record. If this fails to complete normally, then perform a MHU-320AE reset recovery. If FMU_ERRUPDATE completes normally but PROTID is still 0, then there could be a fault on the error wires which set the W bit. The error wires can be disabled by clearing FMU_ERR<n>CTRL.W_EN. Operation can continue but without the error reporting redundancy provided by the error wires and the FMU_ERR<n>STATUS.W bit.

6.3.1 Enable or disable a protection mechanism

All protection mechanisms are enabled on reset.

To enable or disable a protection mechanism, write to the following fields in the FMU_SMEN register:

- BLKTYPE, selects the MHU block type

- BLKID, selects the MHU block
- SMID, selects the specific protection mechanism in the MHU block to be enabled or disabled.

We recommend that software does not disable protection mechanisms.

6.3.2 Enable or disable both error signals on a block

Each block has a critical error signal output and a non-critical error signal output. Software can enable or disable both output signals on a block.

At reset, the MHU-320AE, enables the error wire outputs for the MHU Sender, MHU Receiver, and the FMU blocks.

To enable or disable the block error signals, write to the following fields in the FMU_SMEN register:

- BLKTYPE, selects the type of MHU block such as MHU Sender or MHU Receiver.
- BLKID, selects a block
- SMID, set to 255
- EN, set to:
 - 0, to disable both error signal outputs on the block.
 - 1, to enable both error signal outputs on the block.

Block error signals status

To discover if the block error signals are enabled, software can write to the FMU_SMRD register with the following fields:

- BLKTYPE, selects the type of MHU block such as MHU Sender or MHU Receiver.
- BLKID, selects a block
- SMID, set to 255
- PAGEID, set to 0

The MHU returns the enable status of these signals when software reads FMU_SMRDATA.enable:

- 0 - Both error wire outputs for that block are not enabled.
- 1 - Both error wire outputs for that block are enabled.

6.3.3 Discover the active errors on the block

To discover if a block has some active errors, software can write to an FMU register, to request that the block resends any errors that have not been cleared.

To request that the block resends any active errors that have not been cleared, software can write SMID=255 to any of the following registers:

- FMU_SMEN

- FMU_SMERR
- FMU_SMCR
- FMU_SMWR

When the selected block receives the message, then it resends any errors that have not been cleared.

6.3.4 Inject an error in a protection mechanism

To inject an error into a protection mechanism, write to the FMU_SMERR register.

The FMU_SMERR.BLKTYPE field specifies the MHU-320AE block type, FMU_SMERR.BLKID field specifies the block ID, and FMU_SMERR.SMID field specifies the protection mechanism into which to inject the error.

When a write to FMU_SMERR completes, FMU_STATUS.BUSY remains set to 1 until any resulting updates to FMU_ERR<n>STATUS are complete.

This method injects only one error. The injected errors clear when the error clears in FMU_ERR<n>STATUS.

Software can use the error injection feature to test the software error recovery handler.

6.4 Ping mechanisms

The MHU-320AE uses a CRC ping mechanism to protect internal AXI5-Stream and ACE5-Lite connections.

The AXI5-Stream and ACE5-Lite protection supports a CRC and ping mechanism that software can enable independently.

The value of the `*_stream_protection_type` configuration parameter controls the protection type for that interface as follows:

- 0 - No protection
- 1 - Parity protection. Only suitable for point-to-point connections.
- 2 - CRC protection
- 3 - CRC and parity protection

If CRC protection is enabled, then a ping mechanism also runs in the background with minimal interference to “mission” traffic.

The ping mechanism is a regular check that the connection is working. Every mission packet is eventually followed by a ping packet. If a ping packet does not generate a ping acknowledge packet

from the recipient, a timeout error is raised. Software can then program the ping timeout value in the FMU_TIMEOUT register.

An 8-bit CRC checksum is used to protect against missing or extra packets, and data corruption. The MHU Sender or the MHU Receiver inserts the CRC packet into the stream.

CRC and ping packets try not to interfere with the mission traffic, so initially mission packets take higher priority. To prevent erroneous ping timeouts from occurring, then eventually the CRC and ping packets get a higher priority than a mission packet.

For more information about how to change MHU ping timeout values and how to determine the CRC error type, see the [FMU_SMRDATA](#) and [FMU_SMWDATA](#) registers.

The CRC and ping mechanism adheres to the power state of the “mission” protocol, so when the bus is in low-power state, no CRC or ping packets are sent.

6.5 Software interaction

Software interacts with the FMU during initialization, when handling interrupts, preventing backpressure, and managing power.

Initialization

The initialization routine can iterate over the FMU_ERR<n>FR registers to discover the capabilities of each error record.



All protection mechanisms are enabled on reset, which might lead to errors being logged in the error records. If the system does not support or want to check a particular safety feature, then the software can disable that protection mechanism.

To disable a protection mechanism, write the corresponding block type, block ID, and protection mechanism ID to the FMU_SMEN register.

To analyze the logged errors, read the FMU_ERR<n>STATUS register.

To clear all logged errors, read the FMU_ERR<n>STATUS and write back the V, UE, and OF bits with the same value that is read.

To enable error reporting through either the ERI or CRI, write to FMU_ERR<n>CTLR.UI or FMU_ERR<n>CTLR.CI, respectively.

Interrupt handler

When an interrupt is received, the interrupt handling software identifies the error record ID by reading the FMU_ERRGSR register. The asserted bit[n] indicates that error record n is in error. For more information about the error, read the FMU_ERR<n>STATUS register.

FMU_ERR<n>STATUS.BLKID indicates the BLKID that reported an error, and BLKTYPE indicates the block type. FMU_ERR<n>STATUS.IERR indicates which protection mechanism reported the error.

If more than one error of the same criticality has been reported by this block type to this error record, then FMU_ERR<n>STATUS.OF is set to 1. If there is overflow, the error record retains the protection mechanism ID of the first error.

When the recovery procedure is complete, the error from this error record can be cleared by reading FMU_ERR<n>STATUS and writing back the V and OF bits with the same value that is read. The software then polls for FMU_STATUS.busy==0.

FMU busy

The APB5 port to the FMU is designed not to introduce backpressure by deasserting the pready signal. This design feature prevents software lockup and always keeps the error records accessible.

There are several operations which take multiple clock cycles to complete within the FMU. The FMU frees up the APB5 bus by asserting the pready signal to complete the APB transaction. However, it might still be processing the previous request.

When software writes to one of the following FMU registers, it must poll the register field FMU_STATUS.BUSY==0, before it issues another write to these registers:

- FMU_ERR<n>STATUS
- FMU_SMEN
- FMU_SMERR
- FMU_SMCR
- FMU_SMWR
- FMU_SMRD
- FMU_ERRUPDATE

Power management

Software can power down either the MHU Sender or MHU Receiver by using the procedure described in [Power management](#). However, performing FMU accesses can result in messages being sent to the remote MHU block, which can affect its power down state. Writing to the following registers generates messages to the remote MHU block.

- FMU_ERR<n>STATUS
- FMU_SMEN
- FMU_SMERR
- FMU_SMCR
- FMU_SMWR
- FMU_SMRD
- FMU_ERRUPDATE

6.6 FuSa programmer's view

The FMU contains the functional safety registers.

MHU-320AE uses a separate and independent register block for the FMU programmer's view. Each instance of the FMU has its own register block. For more information on the FMU register block, see [FMU register block summary](#).

6.7 FuSa I/O

The MHU-320AE has extra signals for FuSa fault detection and control.

The following table lists the protection mechanism that MHU-320AE uses for each AMBA interface or signal type.

Table 6-5: AMBA interface FuSa ports

Interface type	Protection mechanism
APB5	AMBA parity
AXI5-Stream interfaces between internal MHU blocks	AMBA parity, or CRC, or both
AXI5-Stream external interfaces: <ul style="list-style-type: none"> Processor interface MSI delivery interface 	AMBA parity
ACE5-Lite	AMBA parity added to all external ACE5-Lite interfaces, including communications interface, when configured to use ACE5-Lite instead of AXI5-Stream.
Q-Channel	AMBA parity
Clock input signal	Duplicated *_chk signal
Reset input signal	Duplicated *_chk signal
Non-AMBA input signal	Odd parity *_chk signal
Non-AMBA output signal	Odd parity *_chk signal
Interrupt outputs	Odd parity *_chk signal
External error interfaces	Odd parity *_chk signal

For more information about the signals, see the *Arm® CoreLink™ MHU-320AE Message Handling Unit Configuration and Integration Manual*.

6.8 External error inputs

Each MHU-320AE block has generic fault inputs that allow the SoC integrator to connect and flag external faults to the FMU.

For example, an SoC integrator might have an external safety mechanism that is physically located next to a MHU block. The SoC integrator can connect the fault signal from this external SM to the `ext_err0_req_*` or `ext_err1_req_*` input signals of the MHU block. If a fault occurs, the MHU flags and reports a fault in the same manner it does with internal faults.

A captured fault is reported in the MHU block error record with a `SM_EXT<n>_<BLKTYPE>` protection name such as `SM_EXT0_SND` or `SM_EXT1_REC`.

6.9 Resets

MHU-320AE uses active-LOW duplicated resets. The two duplicated resets must change on the same clock edge, although MHU-320AE has an allowed skew tolerance after synchronizing the two resets, before a reset error is generated.

MHU-320AE only enters reset when both duplicated resets are asserted.

MHU-320AE exits reset when either duplicated reset deasserts.

The MHU-320AE has a reset synchronizer, so that on reset the post-synchronizer reset signals assert asynchronously and deasserts synchronously.

Although the FMU resides in the either the sender or receiver MHU block, the FMU has separate `fm_u_reset_n` and `fm_u_reset_n_chk` signals. Therefore, the MHU can be reset without resetting the FMU, which enables retention of the FMU error records.

6.9.1 Cold reset sequence

Follow these steps to initiate a Cold reset of MHU-320AE.

Procedure

1. Assert `reset_n` and `reset_n_chk` simultaneously.
The post-synchronizer reset signals, `reset_n_pri` and `reset_n_sec`, assert asynchronously at the same time.
2. Keep the resets asserted for at least 3 cycles more than the number of stages in the reset synchronizer. This reset assertion guarantees a reset flush through the non-resettable flops.
3. Release the resets.
When the `reset_n` or `reset_n_chk` signal deasserts, the `reset_n_pri` signal deasserts synchronously, followed by the `reset_n_sec` signal two clk cycles later. `reset_n` and `reset_n_chk` signals deassert at the same time. .

6.10 RAM protection

RAM instances are not duplicated and are shared by both primary and secondary `noram` instances.

RAMs are protected with the MHU-320AE ECC scheme that distinguishes between:

- SEC errors on the data
- SEC errors on the address
- DED errors
- White noise errors, that is, detection of all 0s or all 1s data

For each RAM, the fault collator allocates a different protection mechanism ID for these RAM error types. Therefore, for the MHU Receiver, the RAM errors are a significant proportion of the protection mechanism ID space.

RAM protection also performs lock-step checking of the RAM address, data, and control outputs for the primary and secondary logic.

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. 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.

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

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 rOp1 identifier indicates the revision status of the product described in this manual, where:

rx	Identifies the major revision of the product.
py	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
0001-03	17 May 2024	Non-Confidential	First EAC release for rOp1
0000-02	15 December 2023	Confidential	First EAC release for rOp0
0000-01	31 October 2023	Confidential	First BET release for rOp0

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 349.

Table 2: Issue 0000-00

Change	Location
First BET release	-

The following table describes the differences between issue 0000-01 and issue 0000-02 of this document.

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

Change	Location
First EAC release	-
Add information for what features are included when Functional safety support is configured	Overview of MHU-320AE
Added more information about the different structure configurations	MHU-320AE structure
Added information about MHU bridge	MHU bridge
Added channels overview section	MHU-320AE channels
Added information about Doorbell channel error recovery procedure	Doorbell channel error recovery procedure
Added information about Fast channel error recovery procedure	Fast channel error recovery procedure
Added sections for RAMs and ECC	RAMs and ECC
Added section on power management	Power management
Added recovery and prevention information about SYN_ACE_CC_BAD	MHU Sender and MHU Receiver software error records, Record 0
Added note about SRAS_ERR0CTL0.DAE and RRAS_ERR0CTL0.DAE register fields	MHU Sender and MHU Receiver software error records, Record 0
Added section about bus errors	Bus errors
Added section about register map pages	Register map pages
Added discovery flow	Discovery flow
Added registers to programmers model	Programmers model for MHU-320AE
Added section about discovering active errors on the block	Discover the active errors on the block
Added section about injecting an error in a protection mechanism	Inject an error in a protection mechanism
Added software interaction power management information for functional safety	Software interaction

The following table describes the differences between issue 0000-02 and issue 0001-03 of this document.

Table 4: Differences between issue 0000-02 and issue 0001-03

Change	Location
Changed product name for MHU-320AE	Throughout the entire document

Change	Location
Replaced references to RASv2 with RASv1.1	<ul style="list-style-type: none"> Features of MHU-320AE Compliance Reliability, Accessibility, and Serviceability Error record classification
Added register type category to register summary	<ul style="list-style-type: none"> MHU FMU registers summary MHU Sender Security Control register summary MHU Sender Postbox registers summary MHU Sender RAS registers summary MHU Receiver Security Control registers summary
Clarified reset information for registers	Throughout the Programmers model for MHU-320AE
Changed reset value for MSIL field of the PBX_FCTRL and MBX_FCTRL register	<ul style="list-style-type: none"> PBX_FCTRL register MBX_FCTRL register
Changed field name from NUM_FCH_PER_GRP to NUM_FCH_PER_FCG for SSC_FCH_CFG0, PBX_FCH_CFG0, RSC_FCH_CFG0, and MBX_FCH_CFG0 registers	<ul style="list-style-type: none"> SSC_FCH_CFG0 PBX_FCH_CFG0 RSC_FCH_CFG0 MBX_FCH_CFG0
Changed field name from flag<x> to MSK<x> in MDBCW<n>_MSK_CLR and MDBCW<n>_MSK_SET registers	<ul style="list-style-type: none"> MDBCW<n>_MSK_CLR MDBCW<n>_MSK_SET

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
<i>italic</i>	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 <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <div>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></div>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . 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.



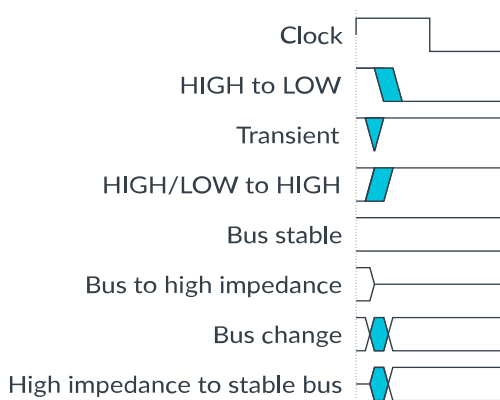
This information reminds you of something important relating to the current content.

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™ MHU-320AE Message Handling Unit Configuration and Integration Manual	107613	Confidential
Arm® CoreLink™ MHU-320AE Message Handling Unit Release Note	109507	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® APB Protocol Specification	IHI 0024D	Non-Confidential
AMBA® AXI Protocol Specification	IHI 0022J	Non-Confidential
AMBA® AXI-Stream Protocol Specification	IHI 0051B	Non-Confidential
AMBA® Low Power Interface Specification	IHI 0068D	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential
Message Handling Unit Architecture version 3.0	AES 0072	Non-Confidential

Non-Arm resources	Document ID	Organization
JEDEC Standard Manufacturer's Identification Code, JEP106	JEP106	www.jedec.org