MPAMF_ESR, MPAM Error Status Register

The MPAMF_ESR characteristics are:

Purpose

Indicates MPAM error status for this MSC.

MPAMF_ESR_s reports Secure MPAM errors. MPAMF_ESR_ns reports Non-secure MPAM errors. MPAMF_ESR_rt reports Root MPAM errors. MPAMF_ESR_rl reports Realm MPAM errors.

Software should write this register after reading the status of an error to reset ERRCODE to `0x0000` and OVRWR to 0 so that future errors are not reported with OVRWR set.

Configuration

This register is present only when FEAT_MPAM is implemented. Otherwise, direct accesses to MPAMF_ESR are res0.

MPAMF_ESR is 64-bit register when MPAM v0.1 or v1.1 is implemented and MPAMF_IDR.HAS_EXTD_ESR == 1.

Otherwise, MPAMF_ESR is a 32-bit register.

If an MSC cannot encounter any of the error conditions listed in 'Errors in MSCs' in ArmÂ® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A (ARM DDI 0598), both the MPAMF_ESR and MPAMF_ECR must be RAZ/WI.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ESR is a:

- 64-bit register when (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_IDR.HAS_EXTD_ESR == 1
- 32-bit register otherwise

Field descriptions

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_IDR.HAS_EXTD_ESR == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																												RIS			
OVRWR	RES0			ERRCODE				PMG								PARTID_MON															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:36]

Reserved, res0.

RIS, bits [35:32]
When MPAMF_IDR.HAS_RIS == 1:

Resource Instance Selector. Where applicable to the ERRCODE, captures the RIS value for the error.

Otherwise:

Reserved, res0.

OVRWR, bit [31]

Overwritten.

If 0 and ERRCODE == `0b0000`, no errors have occurred.

If 0 and ERRCODE is nonzero, a single error has occurred and is recorded in this register.

If 1 and ERRCODE is nonzero, multiple errors have occurred and this register records the most recent error.

The state where this bit is 1 and ERRCODE is zero must not be produced by hardware and is only reached when software writes this combination into this register.

Bits [30:28]

Reserved, res0.

ERRCODE, bits [27:24]

Error code.

ERRCODE	Meaning
`0b0000`	No error.

`0b0001`	PARTID_SEL_Range.
`0b0010`	Req_PARTID_Range.
`0b0011`	MSMONCFG_ID_RANGE.
`0b0100`	Req_PMG_Range.
`0b0101`	Monitor_Range.
`0b0110`	intPARTID_Range.
`0b0111`	Unexpected_INTERNAL.
`0b1000`	Undefined_RIS_PART_SEL.
`0b1001`	RIS_No_Control.
`0b1010`	Undefined_RIS_MON_SEL.
`0b1011`	RIS_No_Monitor.
`0b1100`	Reserved.
`0b1101`	Reserved.
`0b1110`	Reserved.
`0b1111`	Reserved.

PMG, bits [23:16]

Program monitoring group.

Set to the PMG on an error that captures PMG. Otherwise, set to `0x00` on an error that does not capture PMG.

PARTID_MON, bits [15:0]

PARTID or monitor.

Set to the PARTID on an error that captures PARTID.

Set to the monitor index on an error that captures MON.

On an error that captures neither PARTID nor MON, this field is set to 0.

Otherwise:

31	30 29 28	27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
OVRWR	RES0	ERRCODE	PMG	PARTID_MON

OVRWR, bit [31]

Overwritten.

If 0 and ERRCODE == `0b0000`, no errors have occurred.

If 0 and ERRCODE is nonzero, a single error has occurred and is recorded in this register.

If 1 and ERRCODE is nonzero, multiple errors have occurred and this register records the most recent error.

The state where this bit is 1 and ERRCODE is 0 must not be produced by hardware and is only reached when software writes this combination into this register.

Bits [30:28]

Reserved, res0.

ERRCODE, bits [27:24]

Error code.

ERRCODE	Meaning
`0b0000`	No error.
`0b0001`	PARTID_SEL_Range.
`0b0010`	Req_PARTID_Range.
`0b0011`	MSMONCFG_ID_RANGE.
`0b0100`	Req_PMG_Range.
`0b0101`	Monitor_Range.
`0b0110`	intPARTID_Range.
`0b0111`	Unexpected_INTERNAL.
`0b1000`	Reserved.
`0b1001`	Reserved.
`0b1010`	Reserved.
`0b1011`	Reserved.
`0b1100`	Reserved.
`0b1101`	Reserved.
`0b1110`	Reserved.
`0b1111`	Reserved.

PMG, bits [23:16]

Program monitoring group.

Set to the PMG on an error that captures PMG. Otherwise, set to `0x00` on an error that does not capture PMG.

PARTID_MON, bits [15:0]

PARTID or monitor.

Set to the PARTID on an error that captures PARTID.

Set to the monitor index on an error that captures MON.

On an error that captures neither PARTID nor MON, this field is set to `0x0000`.

Accessing MPAMF_ESR

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MPAMF_ESR_s must only be accessible from the Secure MPAM feature page.
- MPAMF_ESR_ns must only be accessible from the Non-secure MPAM feature page.
- MPAMF_ESR_rt must only be accessible from the Root MPAM feature page.
- MPAMF_ESR_rl must only be accessible from the Realm MPAM feature page.

MPAMF_ESR_s, MPAMF_ESR_ns, MPAMF_ESR_rt, and MPAMF_ESR_rl must be separate registers:

- The Secure instance (MPAMF_ESR_s) accesses the error status used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ESR_ns) accesses the error status used for Non-secure PARTIDs.
- The Root instance (MPAMF_ESR_rt) accesses the error status used for Root PARTIDs.
- The Realm instance (MPAMF_ESR_rl) accesses the error status used for Realm PARTIDs.

MPAMF_ESR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	`0x00F8`	MPAMF_ESR_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	`0x00F8`	MPAMF_ESR_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	`0x00F8`	MPAMF_ESR_rt

When FEAT_RME is implemented, accesses on this interface are **RW**.

Component	Frame	Offset	Instance

MPAM	MPAMF_BASE_rl	`0x00F8`	MPAMF_ESR_rl

When FEAT_RME is implemented, accesses on this interface are **RW**.

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94