RISC-V External Debug Security Specification

Version v0.7.3, 2026-02-10: Release Candidate for Internal Review

Table of Contents

Preamble

This document is in the Development state

Expect potential changes. This draft specification is likely to evolve before it is accepted as a standard. Implementations based on this draft may not conform to the future standard.

Copyright and license information

This specification is licensed under the Creative Commons Attribution 4.0 International License (CC-BY 4.0). The full license text is available at creativecommons.org/licenses/by/4.0/.

Copyright 2023-2024 by RISC-V International.

Contributors

This RISC-V specification has been contributed to directly or indirectly by (in alphabetical order): Allen Baum, Aote Jin (editor), Beeman Strong, Erik Kraft, Gokhan Kaplayan, Greg Favor, Iain Robertson, Joe Xie (editor), Paul Donahue, Ravi Sahita, Robert Chyla, Thomas Roecker, Tim Newsome, Ved Shanbhogue, Vicky Goode

Chapter 1. Introduction

Debugging and tracing are essential for developers to identify and rectify software and hardware issues, optimize performance, and ensure robust system functionality. The debugging and tracing extensions in the RISC-V ecosystem play a pivotal role in enabling these capabilities, allowing developers to monitor and control the execution of programs during the development, testing, and production phases. However, the current RISC-V Debug specification grants the external debugger the highest privilege in the system, regardless of the privilege level at which the target system is running, and tracing is also allowed in all privilege modes. This leads to privilege escalation and information leakage issues when multiple actors are present.

This specification defines Debug Module Security Extension (non-ISA extension) and Sdsec (ISA extension) to address the above security issues in the current *The RISC-V Debug Specification* [1] and trace specifications [2] [3].

A summary of the changes introduced by *The RISC-V External Debug Security Specification* follows.

- **Per-Hart Debug Control:** Introduce per-hart states to control whether external debug is allowed for each privilege mode.
- **Per-Hart Trace Control:** Introduce per-hart states to control whether tracing is allowed for each privilege mode.
- **Non-secure debug:** Add a non-secure debug state to relax security constraints for backward compatibility.
- **Debug Mode entry:** An external debugger can only halt the hart and enter Debug Mode when external debug is allowed in the current privilege mode.
- **Memory Access:** Memory access from a hart's point of view, using the Program Buffer or an Abstract Command, must be checked by the hart's memory protection mechanisms as if the hart is running at debug access privilege level; memory access from the Debug Module using System Bus Access (SBA) must be checked by a system memory protection mechanism, such as IOPMP or WorldGuard.
- **Register Access:** Register access using the Program Buffer or an Abstract Command works as if the hart is running at debug access privilege level instead of the M-mode privilege level. The debug CSRs (`dcsr` and `dpc`) are shadowed in lower privilege modes.

1.1. Terminology

Abstract command	A high-level Debug Module operation used to interact with and control harts
Debug Access Privilege	The privilege level with which an Abstract Command or instruction in the Program Buffer accesses hardware resources
Debug Mode	An additional privilege mode to support off-chip debugging
Hart	A RISC-V hardware thread
IOPMP	Input-Output Physical Memory Protection unit
M-mode	The highest privileged mode in the RISC-V privilege model
PMA	Physical Memory Attributes
PMP	Physical Memory Protection unit
Program buffer	A mechanism that allows the Debug Module to execute arbitrary instructions on a hart
Trace encoder	A piece of hardware that takes in instruction execution information from a RISC-V hart and transforms it into trace packets

Chapter 2. External Debug Security Threat Model

Modern SoC development involves several different actors who may not trust each other, resulting in the need to isolate actors' assets during the development and debugging phases. The current RISC-V Debug specification [1] grants external debuggers the highest privilege in the system, regardless of the privilege level at which the target system is running. This leads to privilege escalation issues when multiple actors are present.

For example, the owner of an SoC, who needs to debug their firmware, may be able to use the external debugger to bypass PMP (including PMP lock `pmpcfg`.L=1) and MMU protections, attacking Boot ROM or other SoC creator's assets.

Additionally, the RISC-V privilege architecture supports multiple software entities at different privilege levels that may not trust each other. These entities may be isolated from each other by mechanisms such as PMP/IOPMP or MMU, and may need different debug and trace policies. Since the external debugger is granted the highest privilege in the system, a malicious entity at a lower privilege level could compromise higher privilege software with the external debugger. Similarly, trace output from higher privilege execution could leak sensitive information to entities at lower privilege levels if not properly controlled.

Chapter 3. Sdsec (ISA extension)

This chapter introduces the Sdsec ISA extension, which adds security enhancements to the Sdext/Sdtrig [1] and trace functionality [2] [3]. The Sdsec extension is only applicable when external debug (Sdext/Sdtrig) or trace functionality (E-Trace/N-Trace) is implemented in the system. The extension provides mandatory M-mode external debug and trace control, along with optional extensions for lower privilege modes.

When external debug is implemented, the following external debug control extensions are defined:

- **Smmdedbg (Base)**: Mandatory base extension providing M-mode external debug control
- **Smsdedbg (Optional)**: Extends Smmdedbg to add S/HS-mode external debug control
- **Smvsdedbg (Optional)**: Extends Smmdedbg to add VS-mode external debug control
- **Smudedbg (Optional)**: Extends Smmdedbg to add U-mode and VU-mode external debug control

When trace is implemented, the following trace control extensions are defined:

- **Smmdetrc (Base)**: Mandatory base extension providing M-mode trace control
- **Smsdetrc (Optional)**: Extends Smmdetrc to add S/HS-mode trace control
- **Smvsdetrc (Optional)**: Extends Smmdetrc to add VS-mode trace control
- **Smudetrc (Optional)**: Extends Smmdetrc to add U-mode and VU-mode trace control

The Smmdedbg base extension is mandatory when external debug is implemented. The Smmdetrc base extension is mandatory when trace is implemented. If a system implements both external debug and trace, both base extensions must be implemented. Users can choose to implement some or all of the optional extensions depending on their system requirements. Debug and trace extensions can be implemented independently - for example, a system may implement Smsdedbg without Smsdetrc, or vice versa. However, when implementing extensions for lower privilege modes, all extensions for higher privilege modes of the same type (external debug or trace) must also be implemented. These extensions work together to establish a hierarchical external debug and trace control where external debug access or trace output can be restricted to specific privilege levels.

3.1. External Debug Security Extensions

External debug operations can modify system state and expose sensitive information. The following extensions provide hierarchical privilege-based controls to restrict debug access, preventing unauthorized debugging of higher-privilege software.

When external debug is allowed for a privilege mode, it is allowed for all lower privilege modes as well. If an optional debug extension is not implemented, its control state will be treated as if it were 0; the control state may either be not implemented or read-only 0 in such cases.

3.1.1. Debug Operations

Chapter 3 of *The RISC-V Debug Specification* [1] outlines all mandatory and optional external debug operations. The operations listed below are affected by the Sdsec extension; other operations remain unaffected. In the context of this chapter, **debug operations** refer to those listed below.

Debug operations affected by Sdsec:

- Halting the hart to enter Debug Mode

- Executing the Program Buffer
- Serving abstract commands (Access Register, Access Memory)

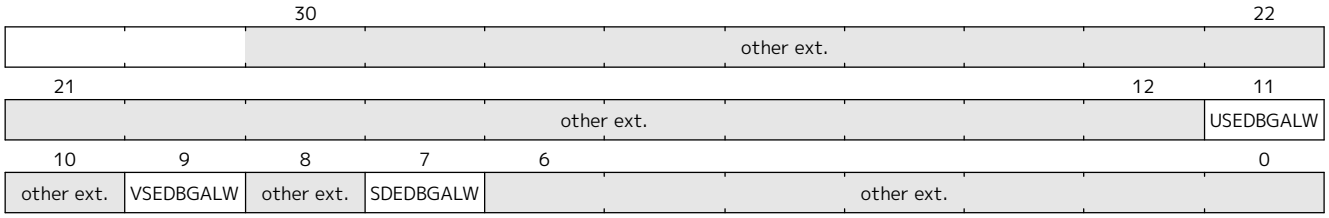
3.1.2. External Debug Security Control Signals/Fields and CSRs

The following table summarizes control signals/fields and associated CSRs introduced by the external debug security extensions.

Table 1. External Debug Control and CSRs

Extension	External Debug Control	New CSRs	Target Modes
Smmdedbg	mdbgen	None	M-mode
Smsdedbg	msdcfg.SDEDBGALW	sdcsr, sdpc	S/HS-mode
Smvsdedbg	msdcfg.VSEDBGALW	None	VS-mode
Smudedbg	msdcfg.USEDBGALW	udcsr, udpc	U-Mode or VU-mode

The `msdcfg` is a machine-mode read/write CSR (address 0x74E) used to hold configurations for supervisor domains. The following fields in `msdcfg` control external debug security. Other fields in `msdcfg` are defined by other extensions. If an optional extension is not implemented, the corresponding field is read-only 0.



Register 1: External debug fields in `msdcfg`

Table 2. Details of external debug fields in `msdcfg`

Field	Description	Access	Reset
SDEDBGALW	S/HS-mode external debug allowed. When set to 1, S/HS-mode external debug is allowed when `mdbgen` is 0. Introduced by Smsdedbg.	WARL	0
VSEDBGALW	VS-mode external debug allowed. When set to 1, VS-mode external debug is allowed when `mdbgen` is 0 and `SDEDBGALW` is 0. Introduced by Smvsdedbg.	WARL	0
USEDBGALW	U-mode/VU-mode external debug allowed. When set to 1, U-mode or VU-mode external debug is allowed when `mdbgen`=0, `SDEDBGALW`=0, and `VSEDBGALW`=0. Introduced by Smudedbg.	WARL	0

The bit positions of `VSEDBGALW` *and* `USEDBGALW` *shown above are tentative and have not been officially allocated. The final bit assignments will be determined during the ratification process.*

3.1.3. Debug Access Privilege

The **debug access privilege** is defined as the privilege level for state accesses from the Debug Module or in the Program Buffer, such as Abstract Commands and instructions in the Program Buffer. Register and memory accesses use the privilege level derived in Table 3. Attempts to access state that requires a privilege level above the **debug access privilege** will fail and set `abstractcs`.CMDERR to 3.

Table 3. External Debug Configuration and Privilege

mdbgen	SDEDBGALW (if implemented)	VSEDBGALW (if implemented)	UEDBGALW (if implemented)	Debug Access Privilege
1	Don't care	Don't care	Don't care	M-mode
0	1	Don't care	Don't care	S/HS-Mode
0	0	1	Don't care	VS-Mode
0	0	0	1	U-Mode or VU-Mode
0	0	0	0	None

3.1.4. Maximum Allowed Resume Privilege Mode

The maximum privilege level that can be configured in `dcsr`.PRV and `dcsr`.V is determined in Section 3.1.4. The fields retain legal values when the `dcsr`.PRV and `dcsr`.V are configured with an illegal privilege level. Illegal privilege levels include unsupported levels and any level higher than the maximum allowed debug privilege.

Table 4. Maximum Allowed Resume Privilege Mode

mdbgen	SDEDBGALW (if implemented)	VSEDBGALW (if implemented)	UEDBGALW (if implemented)	Maximum Debug Allowed Privilege on resume
1	Don't care	Don't care	Don't care	M-Mode
0	1	Don't care	Don't care	S/HS-Mode
0	0	1	Don't care	VS-Mode
0	0	0	1	U-Mode or VU-Mode
0	0	0	0	None

3.1.5. M-mode External Debug Security Extension (Smmdedbg)

The Smmdedbg extension is the mandatory base extension when external debug is implemented.

The `mdbgen` may be controlled through various methods, such as a new input port to the hart, a handshake with the system Root of Trust (RoT), or other methods. The `mdbgen` state for the Root of Trust (RoT) itself should be managed by SoC hardware, likely depending on lifecycle fusing. The implementation can choose to group several harts together and use one signal to drive their `mdbgen` state or assign each hart its own dedicated state. For example, a homogeneous computing system can use a signal to drive all `mdbgen` states to enforce a unified debug policy across all harts.

A state element in each hart, named `mdbgen`, is introduced to control the debuggability of M-mode for each hart. When `mdbgen` is set to 1, debug is allowed for M-mode and the following rules apply:

- The debug access privilege for the hart is M-mode. Abstract Commands, including Quick Access, and instructions in Program Buffer operate with M-mode privilege.
- The debug operations are allowed when the hart executes in any privilege mode.
- Privilege-changing instructions (other than EBREAK) executed in the Program Buffer must either act as a NOP or raise an exception (stopping execution and setting `abstractcs`.CMDERR to 3).
- The maximum allowed resume privilege mode is M-mode.

When `mdbgen` is set to 0 and the hart is running in M-mode, external debug is disallowed for M-mode:

- The hart will not enter Debug Mode:
  - Halt requests will remain pending until external debug is allowed.
  - Triggers with ACTION=1 (enter Debug Mode) will not match or fire.
  - EBREAK cannot enter Debug Mode and always raises a breakpoint exception.
- The external trigger outputs (with ACTION=8/9) will not match or fire.
- Accesses to M-mode accessible CSRs (e.g., `dcsr`, `dpc`, and `dscratch0/1`) will fail and `abstractcs`.CMDERR is set to 3 (exception).
- The configuration in `dcsr` will be ignored as if it were 0.
- DMODE is read/write for accesses from M-mode when Sdtrig is implemented.

M-mode is given write access to DMODE when `mdbgen` *is 0 to allow M-mode software to save and restore trigger context at S/HS-mode software boundaries. This capability is necessary to prevent triggers from serving as a side-channel attack vector against S/HS-mode software where external debugging is disallowed. Without this, an external debugger could indirectly observe states (such as PC and data addresses) from the debug-disallowed software, potentially leading to information leakage or Denial of Service (DoS). By enabling M-mode to switch trigger contexts at privilege mode boundaries, such attacks can be prevented. When* `mdbgen` *is 1, DMODE remains accessible only from Debug Mode.*

If the hart is running in a debug-allowed lower privilege mode (e.g., S/HS-mode) when `mdbgen` is 0:

- Single-stepping cannot stop in M-mode.
- Interrupts to M-mode cannot be disabled by setting `dcsr`.STEPIE=0.

When `mdbgen`*=0 and* `dcsr`*.STEP=1, a single-stepped instruction in a debug-allowed privilege mode may transfer control to the M-mode trap handler. The hart will execute the handler in M-mode and re-enter Debug Mode immediately after an MRET instruction returns to the debug-allowed privilege mode (i.e., MRET with* `mstatus`*.MPP<3). The hart does not re-enter Debug Mode if the MRET instruction returns to a debug-disallowed privilege mode (i.e., MRET with* `mstatus`*.MPP=3,* `mdbgen`*=0).*

This specification assumes that the debugger ensures `mdbgen` *shall never be set to 0 while the hart is in Debug Mode. Setting* `mdbgen` *to 0 while in Debug Mode could lead to undefined behavior; the hart may lose its debug privileges unexpectedly, potentially causing the debug session to fail or become insecure.*

3.1.6. S/HS-mode External Debug Security Extension (Smsdedbg)

The Smsdedbg extension is an optional extension available when Smmdedbg is implemented. It introduces the `SDEDBGALW` field (bit 7) in CSR `msdcfg`, and controls S/HS-mode debuggability when M-mode external debug is disallowed.

The `SDEDBGALW` field only takes effect when `mdbgen` is 0.

When `SDEDBGALW` is set to 1, S/HS-mode external debug is allowed:

- The debug access privilege for the hart is S/HS-mode. Abstract Commands, including Quick Access, and instructions in Program Buffer operate with S/HS-mode privilege.
- The debug operations are allowed when the hart executes in S/HS-mode, U-mode, VS-mode, or VU-mode.

- Privilege-changing instructions (other than EBREAK) executed in the Program Buffer must either act as a NOP or raise an exception (stopping execution and setting `abstractcs`.CMDERR to 3).
- The new `sdcsr` and `sdpc` (Section 3.1.6.1) are introduced in Smsdedbg to access `dcsr` and `dpc` by an S/HS-mode debugger.
- The **effective debug access privilege** of loads and stores in Debug Mode may be modified as described in Section 3.1.6.2.
- The maximum allowed resume privilege mode is S/HS-mode.

When `SDEDBGALW` is set to 0 and the hart is running in S/HS-mode, external debug is disallowed for S/HS-mode:

- The hart will not enter Debug Mode while running in S/HS-mode:
  - Halt requests will remain pending until external debug is allowed.
  - Triggers with ACTION=1 (enter Debug Mode) will not match or fire.
  - EBREAK cannot enter Debug Mode and always raises a breakpoint exception.
- The external trigger outputs (with ACTION=8/9) will not match or fire while in S/HS-mode.
- Accesses to S/HS-mode accessible CSRs (e.g.,`sdcsr`, `sdpc`) will fail and `abstractcs`.CMDERR is set to 3 (exception).
- The configuration visible in `sdcsr` will be ignored as if it were 0.

When `SDEDBGALW` is set to 0 and the hart is running in a debug-allowed lower privilege mode (e.g, U-mode), the following restrictions apply:

- Single-stepping cannot stop in S/HS-mode.
- Interrupts delegated to S/HS-mode cannot be disabled by setting `dcsr`.STEPIE=0.

When `SDEDBGALW`=0 and `sdcsr`.STEP=1, a single-stepped instruction in a debug-allowed privilege mode may transfer control to the S/HS-mode trap handler. The hart will execute the handler in S/HS-mode and re-enter Debug Mode immediately after an SRET instruction returns to the debug-allowed privilege mode (i.e., SRET with `sstatus`.SPP=0). The hart does not re-enter Debug Mode if the SRET instruction returns to a debug-disallowed privilege mode.

`sdcsr` and `sdpc`

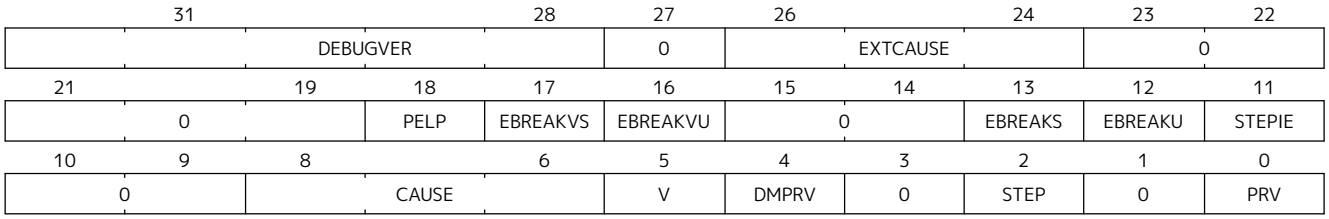
When `SDEDBGALW` is 1, the `sdcsr` and `sdpc` registers provide S/HS-mode read/write access to the `dcsr` and `dpc` registers respectively. However, `sdcsr` does not expose access to the `MPRVEN` field; instead, it repurposes the `MPRVEN` bit position with a `DMPRV` field to modify the **effective debug access privilege** in S/HS-mode. Both registers are only accessible in Debug Mode.

Table 5. Allocated addresses for S/HS-mode shadow of Debug Mode CSR

Number	Name	Description
Oxaaa	sdcsr	S/HS-mode debug control and status register.
Oxaaa	sdpc	S/HS-mode debug program counter.

The `sdcsr` register exposes a subset of `dcsr`, formatted as shown in Register 2, while the `sdpc` register provides full access to `dpc`.

Unlike `dcsr` and `dpc`, the `dscratch0/1` registers do not have a S/HS-mode access mechanism, and external debuggers with S/HS-mode privilege cannot use them.



Register 2: S/HS-mode debug control and status register (sdcsr)

The NMIP, MPRVEN, STOPTIME, STOPCOUNT, EBREAKM, and CETRIG fields in `dcsr` *are configurable only by M-mode; they are masked in* `sdcsr`*, while PRV[1] is hardwired to 0 in* `sdcsr`*. The field for MPRVEN is reclaimed by DMPRV in the* `sdcsr` *layout to avoid wasting fields.*

Table 6. Details of the `dmprv` *field in* `sdcsr`

Field	Description	Access	Reset
DMPRV	0 (normal): The privilege level in Debug Mode is not modified. 1: In Debug Mode, the privilege level for load and store operations is modified to the effective debug access privilege as described in Section 3.1.6.2 and Section 3.1.7.2.	WARL	0

Debug Access Privilege to memory in Smsdedbg

The `sdcsr`.DMPRV takes effect when `mdbgen` is 0, and it is read-only 0 when `mdbgen` is 1. With `SDEDBGALW` set to 1, the **effective debug access privilege** of loads and stores by an S/HS-mode debugger to access memory in Debug Mode can be modified by `sdcsr`.DMPRV. When `sdcsr`.DMPRV=0, the **effective debug access privilege** of loads and stores in Debug Mode follows Table 3; when `sdcsr`.DMPRV=1, the **effective debug access privilege** of loads and stores in Debug Mode is represented by `sstatus`.SPP and `hstatus`.SPV (if the hypervisor extension is supported).

The `sdcsr`.DMPRV does not affect the virtual-machine load/store instructions, HLV, HLVX, and HSV.

3.1.7. VS-mode External Debug Security Extension (Smvsdedbg)

The Smvsdedbg extension is an optional extension available when Smsdedbg is implemented. It introduces the `VSEDBGALW` field (bit TBD) in CSR `msdcfg`, and controls VS-mode debuggability when S/HS-mode external debug is disallowed.

The `VSEDBGALW` field only takes effect when both `mdbgen` is 0 and `SDEDBGALW` is 0.

When `VSEDBGALW` is set to 1, VS-mode external debug is allowed:

- The debug access privilege for the hart is VS-mode. Abstract Commands, including Quick Access, and instructions in Program Buffer operate with VS-mode privilege.
- The debug operations are allowed when the hart executes in VS-mode or VU-mode.
- Privilege-changing instructions (other than EBREAK) executed in the Program Buffer must either act as a NOP or raise an exception (stopping execution and setting `abstractcs`.CMDERR to 3).
- The `sdcsr` and `sdpc` (from Smsdedbg) provide VS-mode read/write access to `dcsr` and `dpc` registers respectively as specified in Section 3.1.7.1.
- The **effective debug access privilege** of loads and stores in Debug Mode may be modified as described in Section 3.1.7.2.

- The maximum allowed resume privilege mode is VS-mode.

When `VSEDBGALW` is set to 0 and the hart is running in VS-mode, external debug is disallowed for VS-mode:

- The hart will not enter Debug Mode while running in VS-mode:
  - Halt requests will remain pending until external debug is allowed.
  - Triggers with ACTION=1 (enter Debug Mode) will not match or fire.
  - EBREAK cannot enter Debug Mode and always raises a breakpoint exception.
- The external trigger outputs (with ACTION=8/9) will not match or fire while in VS-mode.
- Accesses to VS-mode accessible CSRs (e.g.,`sdcsr`, `sdpc`) will fail and `abstractcs`.CMDERR is set to 3 (exception).
- The VS-mode visible configuration in `sdcsr` will be ignored as if it were 0.

When `VSEDBGALW` is set to 0 and the hart is running in a debug-allowed VU-mode, the following restrictions apply:

- Single-stepping cannot stop in VS-mode.
- Interrupts delegated to VS-mode cannot be disabled by setting `sdcsr`.STEPIE=0.

When `VSEDBGALW`=0 and `sdcsr`.STEP=1, a single-stepped instruction in a debug-allowed VU-mode may transfer control to the VS-mode trap handler. The hart will execute the handler in VS-mode and re-enter Debug Mode immediately after an SRET instruction returns to the debug-allowed VU-mode. The hart does not re-enter Debug Mode if the SRET instruction returns to a debug-disallowed VS-mode.

`sdcsr` and `sdpc` in VS-mode

When `VSEDBGALW` is 1, the `sdcsr` and `sdpc` (Section 3.1.6.1) are accessible with VS-mode privilege, providing access to the `dcsr`. The `sdcsr`.EBREAKS and `sdcsr`.EBREAKU fields are redirected to `dcsr`.EBREAKVS and `dcsr`.EBREAKVU, while writes to `sdcsr`.EBREAKVS, `sdcsr`.EBREAKVU, and `sdcsr`.V are discarded (writes are ignored and reads return 0). Similar to `sdcsr` access when `SDEDBGALW` is 1, `sdcsr`.DMPRV modifies the **effective debug access privilege** in VS-mode.

Redirected access to `dcsr`.EBREAKVS and `dcsr`.EBREAKVU unifies the configuration for both S/HS-mode and VS-mode. However, the virtualization mode cannot be changed through `sdcsr`.V by a VS-mode debugger.

Debug Access Privilege to memory in Smvsdedbg

The `sdcsr`.DMPRV modifies the **effective debug access privilege** of loads and stores for a VS-mode debugger when `SDEDBGALW` is 0 and `VSEDBGALW` is 1.

When `sdcsr`.DMPRV=0, the **effective debug access privilege** of loads and stores in Debug Mode follows Table 3; when `sdcsr`.DMPRV=1, the **effective debug access privilege** of loads and stores in Debug Mode is represented by `vsstatus`.SPP with the virtualization mode being honored as 1.

3.1.8. U-mode and VU-mode External Debug Security Extension (Smudedbg)

The Smudedbg extension is an optional extension available when Smmdedbg or Smsdedbg or Smvsdedbg is implemented. It introduces the `USEDBGALW` field (bit TBD) in CSR `msdcfg`, and controls U-mode or VU-

mode (when virtualization mode is 1) debuggability when M-mode or S/HS-mode or VS-mode external debug are disallowed.

The `USEDBGALW` field only takes effect when both `mdbgen`=0 and `SDEDBGALW`=0 and `VSEDBGALW`=0.

When `USEDBGALW` is set to 1, U-mode or VU-mode (when virtualization mode is 1) external debug is allowed:

- The debug access privilege for the hart is U-mode or VU-mode. Abstract Commands, including Quick Access, and instructions in Program Buffer operate with U-mode or VU-mode privilege.
- The debug operations are allowed when the hart executes in U-mode or VU-mode.
- Privilege-changing instructions (other than EBREAK) executed in the Program Buffer must either act as a NOP or raise an exception (stopping execution and setting `abstractcs`.CMDERR to 3).
- The `udcsr` and `udpc` are introduced in Smudedbg to provide U-mode or VU-mode read/write access to `dcsr` and `dpc` registers respectively, as specified in Section 3.1.8.1.
- The maximum allowed resume privilege mode is U-mode or VU-mode.

When `USEDBGALW` is set to 0, external debug is disallowed for all modes:

- The hart will not enter Debug Mode
  - Halt requests will remain pending until external debug is allowed.
  - Triggers with ACTION=1 (enter Debug Mode) will not match or fire.
  - EBREAK cannot enter Debug Mode and always raises a breakpoint exception.
- The external trigger outputs (with ACTION=8/9) will not match or fire.
- Accesses to CSRs without halting will fail and `abstractcs`.CMDERR is set to 3 (exception).
- The configuration in `dcsr` will be ignored as if it were 0.

Software in a higher privilege mode is responsible for managing the debug policy of U-mode or VU-mode and must ensure that a debug-allowed VU-mode cannot be exploited to breach a debug-disallowed U-mode, or vice versa.

`udcsr` and `udpc`

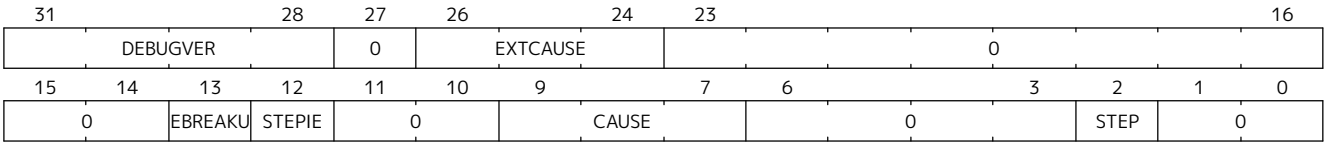
The `udcsr` and `udpc` registers provide U-mode or VU-mode read/write access to the `dcsr` and `dpc` registers respectively. The `udcsr` exposes a subset of `dcsr` and the accessible fields are listed in Register 3. The read/write access to the `udcsr`.EBREAKU field is redirected to `dcsr`.EBREAKVU when the virtualization mode is 1. The `udpc` register provides full access to `dpc`.

Redirected access to `dcsr`.EBREAKVU unifies the configuration for both U-mode and VU-mode.

Table 7. Allocated addresses for U-mode shadow of Debug Mode CSR

Number	Name	Description
Oxaaa	udcsr	U-mode debug control and status register.
Oxaaa	udpc	U-mode debug program counter.

The `udcsr` register exposes a subset of `dcsr`, formatted as shown in Register 3, while the `udpc` register provides full access to `dpc`.



Register 3: U-mode debug control and status register (udcsr)

3.2. Trace Security Extensions

When trace is implemented, trace output can expose sensitive information across privilege boundaries. The extensions below provide security controls to restrict trace output based on privilege levels.

When trace is allowed for a privilege mode, it is allowed for all lower privilege modes as well. If an optional trace extension is not implemented, its control state will be treated as if it were 0; the control state may either be not implemented or read-only 0.

The availability of trace output is controlled through signals defined in the hart-trace interface (HTI) [4]. The logical `sec_inhibit` *signal can be converted to the canonical trace interface signals by implementation.*

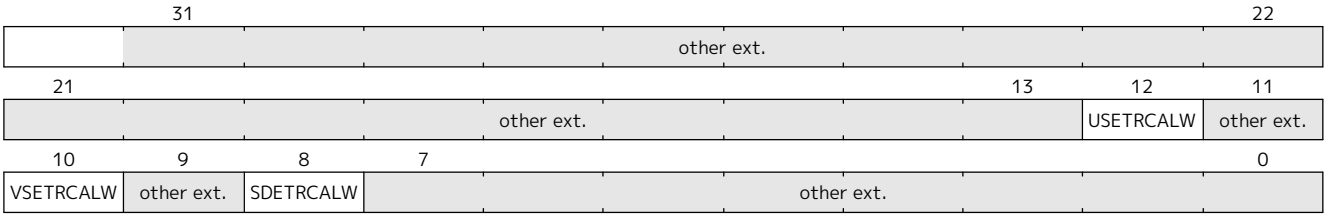
3.2.1. Trace Security Control Signals/Fields

The following tables summarize control signals/fields introduced by the trace security extensions.

Table 8. Trace Control Signals/Fields

Extension	Trace Control	Target Modes
Smmdetrc	mtrcen	M-mode
Smsdetrc	msdcfg.SDETRCALW	S/HS-mode
Smvsdetrc	msdcfg.VSETRCALW	VS-mode
Smudetrc	msdcfg.USETRCALW	U-Mode or VU-mode

The following fields in `msdcfg` control trace security. Other fields in `msdcfg` are defined by other extensions. If an optional extension is not implemented, the corresponding field is read-only 0.



Register 4: Trace security fields in `msdcfg`

Table 9. Details of trace security fields in `msdcfg`

Field	Description	Access	Reset
SDETRCALW	S/HS-mode external trace allowed. When set to 1, S/HS-mode trace is allowed when `mtrcen` is 0. Introduced by Smsdetrc.	WARL	0
VSETRCALW	VS-mode external trace allowed. When set to 1, VS-mode trace is allowed when `mtrcen` is 0 and `SDETRCALW` is 0. Introduced by Smvsdetrc.	WARL	0

Field	Description	Access	Reset
USETRCALW	U-mode/VU-mode external trace allowed. When set to 1, U-mode or VU-mode trace is allowed when `mtrcen`=0, `SDETRCALW`=0, and `VSETRCALW`=0. Introduced by Smudetrc.	WARL	0

The bit positions of `VSETRCALW` *and* `USETRCALW` *shown above are tentative and have not been officially allocated. The final bit assignments will be determined during the ratification process.*

3.2.2. M-mode Trace Security Extension (Smmdetrc)

The Smmdetrc extension is the mandatory base extension when trace is implemented. A state element in each hart, named `mtrcen`, is introduced to control trace output for M-mode.

The `mtrcen` *may be controlled through various methods, such as a new input port to the hart, a handshake with the system Root of Trust (RoT), or other methods. The* `mtrcen` *state for the Root-of-Trust (RoT) itself should be managed by SoC hardware, likely dependent on lifecycle fusing. The implementation can choose to group several harts together and use one signal to drive their* `mtrcen` *state or assign each hart its own dedicated state. For example, a homogeneous computing system can use a signal to drive all* `mtrcen` *states to enforce a unified trace policy across all harts.*

When `mtrcen` is set to 1, trace is allowed for all privilege modes. When `mtrcen` is set to 0, trace is inhibited for M-mode by asserting the logical `sec_inhibit` signal to the trace encoder when the hart is executing in M-mode.

3.2.3. S/HS-mode Trace Security Extension (Smsdetrc)

The Smsdetrc extension is an optional extension available when Smmdetrc is implemented. It introduces the `SDETRCALW` field (bit 8) in CSR `msdcfg`, and controls S/HS-mode trace output when M-mode trace is disallowed.

The `SDETRCALW` field only takes effect when `mtrcen` is 0.

When `SDETRCALW` is set to 1, trace is allowed for all privilege modes except M-mode. When `SDETRCALW` is set to 0, trace is inhibited for S/HS-mode by asserting the logical `sec_inhibit` signal to the trace encoder when the hart is executing in S/HS-mode.

3.2.4. VS-mode Trace Security Extension (Smvsdetrc)

The Smvsdetrc extension is an optional extension available when Smsdetrc is implemented. It introduces the `VSETRCALW` field (bit TBD) in CSR `msdcfg`, and controls VS-mode trace output when S/HS-mode trace is disallowed.

The `VSETRCALW` field only takes effect when both `mtrcen` is 0 and `SDETRCALW` is 0.

When `VSETRCALW` is set to 1, trace is allowed for VS-mode and VU-mode. When `VSETRCALW` is set to 0, trace is inhibited for VS-mode by asserting the logical `sec_inhibit` signal to the trace encoder when the hart is executing in VS-mode.

3.2.5. U-mode and VU-mode Trace Security Extension (Smudetrc)

The Smudetrc extension is an optional extension available when Smmdetrc or Smsdetrc or Smvsdetrc is implemented. It introduces the `USETRCALW` field (bit TBD) in CSR `msdcfg`, and controls U-mode or VU-mode (when virtualization mode is 1) trace output when M-mode or S/HS-mode or VS-mode trace is disallowed.

The `USETRCALW` field only takes effect when both `mtrcen`=0 and `SDETRCALW`=0 and `VSETRCALW`=0.

When `USETRCALW` is set to 1, trace is allowed for U-mode or VU-mode. When `USETRCALW` is set to 0, trace is inhibited for all privilege modes by asserting the logical `sec_inhibit` signal to the trace encoder.

Chapter 4. Debug Module Security (non-ISA) Extension

This chapter defines the required security enhancements for the Debug Module. The debug operations listed below are modified by this extension.

- Halt
- Reset
- Keepalive
- Abstract commands (Access Register, Quick Access, Access Memory)
- System bus access

If any hart in the system implements the Sdsec extension, the Debug Module must also implement the Debug Module Security Extension.

4.1. External Debug Security Extensions Discovery

The ISA and non-ISA external debug security extensions impose security constraints and introduce non-backward-compatible changes. The presence of the extensions can be determined by polling the ALLSECURED and/or ANYSECURED bits in `dmstatus` Table 10. These bits will read as 0 when `nsecdbg` is set to 1, regardless of whether the harts implement the Sdsec extension. When `nsecdbg` is 0, ALLSECURED is set to 1 if all currently selected harts implement the Sdsec extension, and ANYSECURED is set to 1 if any currently selected hart implements the Sdsec extension. When any hart implements the Sdsec extension, it indicates that the Debug Module implements the Debug Module Security Extension as described in this chapter.

4.2. Halt

The halt behavior for a hart is detailed in Section 3.1. According to *The RISC-V Debug Specification* [1], a halt request must be responded to within one second. However, this constraint must be relaxed, as the request might remain pending when debugging is disallowed. In the case of a halt-on-reset request (SETRESETHALTREQ), the request is only acknowledged by the hart once it has reached a privilege level in which debug is allowed.

4.3. Reset

The HARTRESET operation resets selected harts. When M-mode debugging is disallowed, the hart will raise a security fault error to the Debug Module on HARTRESET operations. The error can be detected through ALLSECFAULT and ANYSECFAULT in `dmstatus`.

The NDMRESET operation is a system-level reset not tied to hart privilege levels and resets the entire system (excluding the Debug Module). The Debug Module Security Extension makes the NDMRESET field read-only 0 when `nsecdbg` is 0. The debugger can determine support for the NDMRESET operation by setting the field to 1 and subsequently verifying the returned value upon reading.

4.4. Keepalive

The SETKEEPALIVE bit serves as an optional request for the hart to remain available for debugging. This bit only takes effect when M-mode debugging is allowed; otherwise, the hart behaves as if the bit is not set.

4.5. Abstract Commands

The hart's response to abstract commands is detailed in Section 3.1. The following subsection delineates the constraints when the Debug Module issues an abstract command.

4.5.1. Relaxed Permission Check `relaxedpriv`

The `relaxedpriv` field is hardwired to 0.

4.5.2. Address Translation AAMVIRTUAL

The field `command`.AAMVIRTUAL determines whether the Access Memory command uses a physical or virtual address. When `mdbgen`=1, the behavior follows the original RISC-V Debug Specification [1]. When `mdbgen`=0:

- If AAMVIRTUAL=0, the hart responds with a security fault error (setting `abstractcs`.CMDERR to 6).
- If AAMVIRTUAL=1, addresses are treated as virtual and are translated and protected according to the **debug access privilege**, as defined in Section 3.1.3, and may be modified to the **effective debug access privilege** as defined in Section 3.1.6.2 or Section 3.1.7.2.

4.5.3. Quick Access

When M-mode debugging is disallowed (`mdbgen`=0) for a hart, any Quick Access operation will be discarded by the Debug Module, causing `abstractcs`.CMDERR to be set to 6.

Quick Access abstract commands initiate a halt, execute the Program Buffer, and resume the selected hart. These halts cannot remain pending until debugging is allowed because the debugger blocks while waiting for completion. To address this, the hart would need to distinguish between Quick Access and other halt requests. Since Quick Access is an optional optimization, the Debug Module Security Extension avoids this additional hardware complexity by disallowing Quick Access when `mdbgen` *is 0.*

4.6. System Bus Access

The System Bus Access must be checked by bus initiator protection mechanisms such as IOPMP [5] or WorldGuard [6]. The bus protection unit can return an error to the Debug Module on illegal accesses; in that case, the Debug Module will set `sbcs`.SBERROR to 6 (security fault error).

Trusted entities like the RoT should configure IOPMP or equivalent protection before external debug is allowed.

4.7. Security Fault Error Reporting

A dedicated error code, security fault error (CMDERR 6), is included in `abstractcs`.CMDERR. Issuance of abstract commands under disallowed circumstances sets `abstractcs`.CMDERR to 6. Additionally, the bus security fault error (SBERROR 6) is introduced in `sbcs`.SBERROR to denote errors related to system bus access.

Error status bits are internally maintained for each hart. The ALLSECFAULT and ANYSECFAULT fields in `dmstatus` indicate the error status of the currently selected harts. These error statuses are sticky and can only be cleared by writing 1 to `dmcs2`.ACKSECFAULT Table 11.

4.8. Non-secure Debug

The state element `nsecdbg` is introduced to retain full debugging capabilities, as if the extensions in this specification were not implemented. When `nsecdbg` is set to 1:

- All debug operations are executed with M-mode privilege (equivalent to having `mdbgen` set to 1) for all harts in the system.
- The NDMRESET operation is allowed.
- The RELAXEDPRIV field may be configurable.
- System Bus Access may bypass the bus initiator protections.
- Trace output is enabled in all privilege modes.

4.9. Update of Debug Module Registers

The Debug Module Security Extension introduces new fields in the Debug Module registers. In `dmstatus`, ANYSECURED and ALLSECURED are added at bit 20 and bit 21 respectively, while ANYSECFAULT and ALLSECFAULT are added at bit 25 and bit 26. The ACKSECFAULT field is added to `dmcs2` at bit 12.

Table 10. Details of newly introduced fields in `dmstatus`

Field	Description	Access	Reset
ALLSECURED	The field is 0 when `nsecdbg` is 1. When `nsecdbg` is 0, this field is 1 if all currently selected harts implement the Sdsec extension.	R	-
ANYSECURED	The field is 0 when `nsecdbg` is 1. When `nsecdbg` is 0, this field is 1 if any currently selected hart implements the Sdsec extension.	R	-
ALLSECFAULT	The field is 1 when all currently selected harts have raised a security fault due to reset or keepalive operation	R	-
ANYSECFAULT	The field is 1 when any currently selected hart has raised a security fault due to reset or keepalive operation	R	-

Table 11. Detail of ACKSECFAULT in `dmcs2`

Field	Description	Access	Reset
ACKSECFAULT	0 (nop): No effect. 1 (ack): Clears error status bits for any selected harts.	W1	-

Appendix A: Theory of Operation

This chapter explains the theory of operation for the External Debug Security Extension. The subsequent diagrams illustrate the reference implementation of security control for external debug and trace, respectively.

A.1. External Debug Security Control

The dedicated debug security policy for a hart is enforced through a hierarchical control model using M-mode control state `mdbgen` and optional lower privilege mode control fields (`SDEDBGALW`, `VSEDBGALW`, `USEDBGALW`) in the `msdcfg` CSR. The `mdbgen` state can be implemented as an input signal to the hart, managed through platform-specific mechanisms such as Debug Module registers, Root of Trust (RoT) handshake, or lifecycle fuses. Moreover, the platform state `nsecdbg` can be used to override the debug security policy at the platform level and provide unrestricted debug access for all privilege modes.

The debug security control logic validates all debug requests and triggers (with ACTION=1) against the current privilege level of the hart and the configured debug control states. The validation procedure involves two steps: 1) checking the platform state `nsecdbg` to determine if unrestricted debug access is allowed; 2) checking the hierarchical controls to determine if external debug is allowed at the hart's current privilege level. Debug requests that fail validation will either be dropped or kept pending until the hart transitions to a debug-allowed privilege mode.

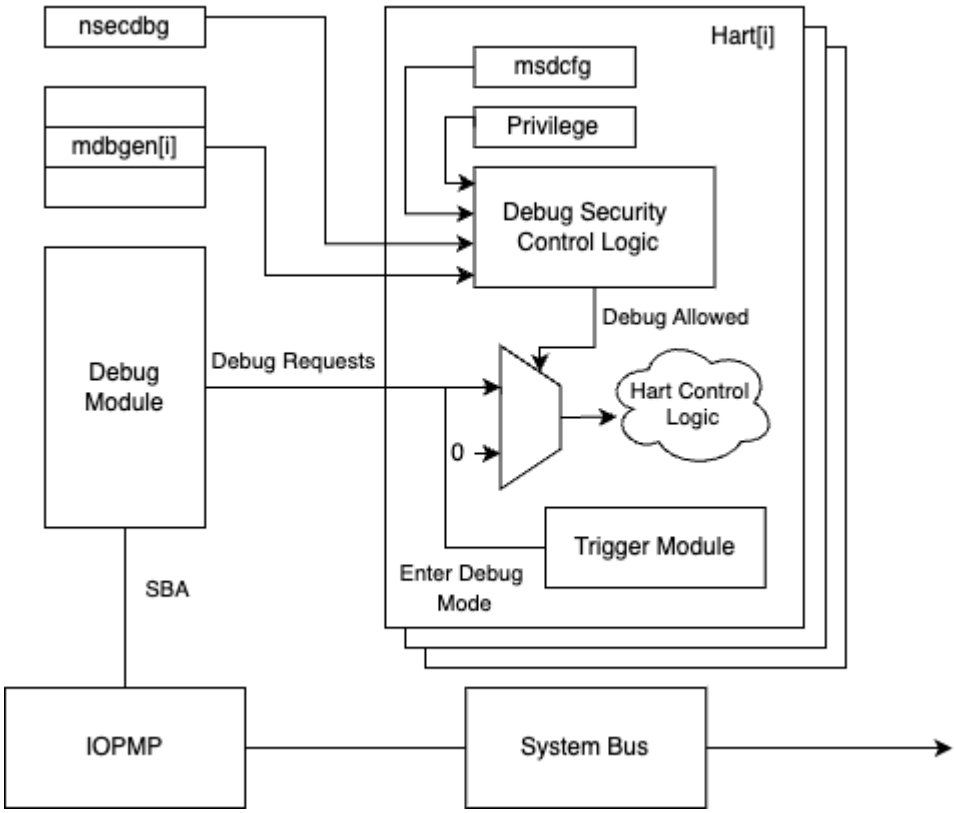


Figure 1. The debug security control

The hierarchical control applies as follows:

- When `mdbgen`=1, external debug is allowed for all privilege modes
- When `mdbgen`=0 and `SDEDBGALW`=1 (if implemented), external debug is allowed for S/HS-mode and lower privilege modes
- When `mdbgen`=0, `SDEDBGALW`=0, and `VSEDBGALW`=1 (if implemented), external debug is allowed for VS-mode and VU-mode

- When all higher privilege controls are 0 and `USEDBGALW`=1 (if implemented), external debug is allowed for U-mode or VU-mode (when virtualization mode is 1) only
- When all control fields are 0, external debug is completely disallowed

The table Table 12 below shows valid implementation combinations for different architectures for external debug:

Table 12. Valid External Debug Extension Combinations

Architecture	Valid Debug Extension Combinations
M-only	• Smmdedbg
M/U	• Smmdedbg • Smmdedbg + Smudedbg
M/S/U	• Smmdedbg • Smmdedbg + Smsdedbg • Smmdedbg + Smsdedbg + Smudedbg
M/S/VS/VU/U	• Smmdedbg • Smmdedbg + Smsdedbg • Smmdedbg + Smsdedbg + Smvsdedbg • Smmdedbg + Smsdedbg + Smvsdedbg + Smudedbg

The debug security control logic must ensure that both the validation and execution of debug requests occur under the same privilege level and the same debug control states to prevent Time-of-Check Time-of-Use (TOCTOU) vulnerabilities. Failing to do so may allow debug requests to bypass security controls.

Self-hosted debugging is also commonly used in application-level debugging, providing a pure software-based debugging solution without the need for an external debugger. It is orthogonal to the external debugging and will not be affected by the debug security controls.

A.2. Trace Security Control

The trace security policy for a hart is enforced through a hierarchical control model using M-mode control state `mtrcen` and optional lower privilege mode control fields (`SDETRCALW`, `VSETRCALW`, `USETRCALW`) in the `msdcfg` CSR. The `mtrcen` state can be implemented as an input signal to the hart, managed through platform-specific mechanisms such as Root of Trust (RoT) handshake or lifecycle fuses.

The security control logic asserts the `sec_inhibit` sideband signal to the trace encoder based on the current privilege level of the hart and the configured trace security control states. The validation procedure checks the hierarchical controls to determine if trace is allowed at the hart's current privilege level. Trace will be inhibited by asserting the `sec_inhibit` signal if the validation fails.

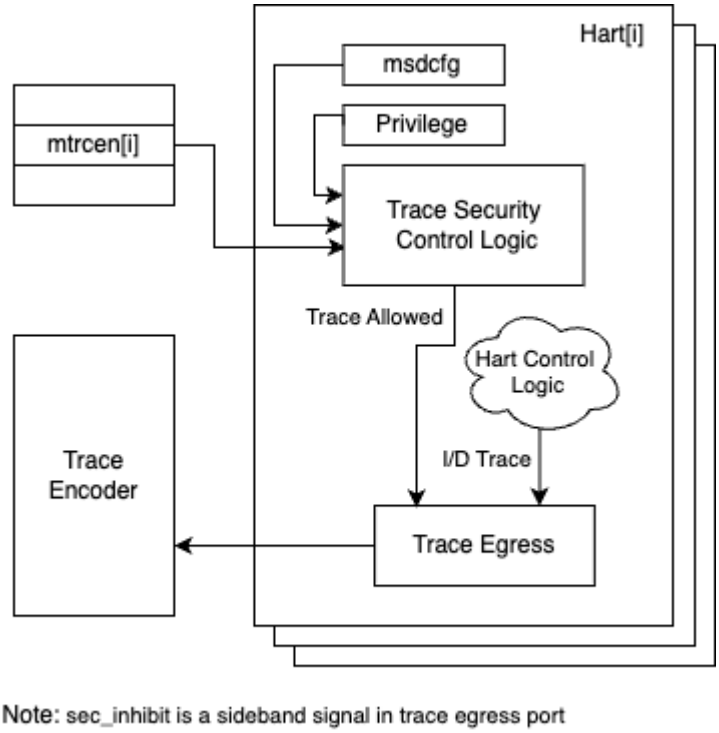


Figure 2. The trace security control

The hierarchical control applies as follows:

- When `mtrcen`=1, trace is allowed for all privilege modes
- When `mtrcen`=0 and `SDETRCALW`=1 (if implemented), trace is allowed for S/HS-mode and lower privilege modes; trace is inhibited for M-mode
- When `mtrcen`=0, `SDETRCALW`=0, and `VSETRCALW`=1 (if implemented), trace is allowed for VS-mode and VU-mode; trace is inhibited for M-mode and S/HS-mode
- When all higher privilege controls are 0 and `USETRCALW`=1 (if implemented), trace is allowed for U-mode or VU-mode (when virtualization mode is 1) only; trace is inhibited for all higher privilege modes
- When all control fields are 0, trace is completely inhibited for all privilege modes

The table Table 13 below shows valid implementation combinations for different architectures for trace:

Table 13. Valid Trace Extension Combinations

Architecture	Valid Trace Extension Combinations
M-only	• Smmdetrc
M/U	• Smmdetrc • Smmdetrc + Smudetrc
M/S/U	• Smmdetrc • Smmdetrc + Smsdetrc • Smmdetrc + Smsdetrc + Smudetrc
M/S/VS/VU/U	• Smmdetrc • Smmdetrc + Smsdetrc • Smmdetrc + Smsdetrc + Smvsdetrc • Smmdetrc + Smsdetrc + Smvsdetrc + Smudetrc

Appendix B: Software Debug/Trace Security Policy Management

This section provides guidance on how software at different privilege levels can manage debug/trace security policies.

B.1. M-mode Management

M-mode software is responsible for managing the `SDEDBGALW`, `VSEDBGALW`, and `USEDBGALW` fields in the `msdcfg` CSR for external debug, and the `SDETRCALW`, `VSETRCALW`, and `USETRCALW` fields in the `msdcfg` CSR for trace. The `mdbgen` and `mtrcen` states are typically controlled by platform hardware (RoT, lifecycle fuses) and are read-only from a software perspective.

M-mode software should:

1. Initialize debug/trace security policy for lower privilege modes based on security requirements
2. Provide interfaces for dynamic debug/trace security policy updates (e.g., through SBI calls from lower privilege modes)
3. Context-switch debug/trace security policies when switching between supervisor software
4. Save and restore trigger context (using DMODE access when `mdbgen`=0) during supervisor software context switches to prevent side-channel attacks

B.2. S/HS-mode Management

S/HS-mode software is responsible for managing debug/trace security policy for lower privilege modes:

1. Call M-mode to apply the debug/trace security policy before launching an application or VM.
2. Provide interfaces to handle dynamic debug/trace security policy update requests (e.g., SBI calls from VMs) and call M-mode to apply the security policy accordingly.
3. Context-switch debug/trace security policies when switching between applications or VMs

B.3. VS-mode Management

VS-mode software is responsible for managing debug/trace security policy for applications in VU-mode:

1. Call S/HS-mode to apply the debug/trace security policy before launching an application.
2. Context-switch debug/trace security policies when switching between applications

Appendix C: Execution Based Implementation with Sdsec

In an execution-based implementation, the code executing the "park loop" can always run with M-mode privilege to access memory and CSRs. However, once execution is dispatched to an Abstract Command or the Program Buffer, the privilege level for accessing memory and CSR should be restricted to debug access privilege.

To achieve this, a Debug Mode only state element (e.g., a field in a custom CSR) may be introduced to control the privilege level in Debug Mode. When the state is set to 1, Debug Mode allows M-mode privilege; when cleared to 0, it enforces the debug access privilege. The hardware sets this state to 1 upon entering the park loop and clears it to 0 by the final instruction of the park loop, right before execution is transferred to an Abstract Command or the program buffer.

Bibliography

[1] “RISC-V Debug Specification.” [Online]. Available: github.com/riscv/riscv-debug-spec.

[2] “RISC-V Efficient Trace for RISC-V.” [Online]. Available: github.com/riscv-non-isa/riscv-trace-spec.

[3] “RISC-V N-Trace (Nexus-based Trace) Specification.” [Online]. Available: github.com/riscv-non-isa/tg-nexus-trace.

[4] “RISC-V Hart Trace Interface.” [Online]. Available: github.com/riscv-non-isa/hart-trace-interface/.

[5] “RISC-V IOPMP Architecture Specification.” [Online]. Available: github.com/riscv-non-isa/iopmp-spec/releases.

[6] “WorldGuard Specification.” [Online]. Available: github.com/riscv-admin/security/blob/main/papers/worldguard%20proposal.pdf.