RISC-V External Debug Security Specification

Version v0.7.2, 2025-11-18: Draft

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

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. It leads to privilege escalation 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 in each privilege mode.
- **Per-Hart Trace Control:** Introduce per-hart states to control whether tracing is allowed in 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 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 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 M-mode privilege level. The debug CSRs (`dcsr` and `dpc`) are shadowed in lower privilege modes, while Smtdeleg/Sstcfg [4] extensions expose the trigger CSRs to S-mode.

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
Supervisor domain	An isolated supervisor execution context defined in RISC-V Supervisor Domains Access Protection [5]

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), MMU, and attack Boot ROM or other firmware (the SoC creator's asset).

Additionally, the RISC-V privilege architecture supports multiple software entities at different privilege levels that do 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, it defines the following external debug control extensions:

- **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, it defines the following trace control extensions:

- **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 model where external debug access or trace output can be restricted to specific privilege levels, preventing unauthorized debugging or tracing.

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 domains.

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. Debug Access Privilege

The **debug access privilege** is defined as the privilege level for state accesses in Debug Mode, such as Abstract Commands and Program Buffer execution. Debug Mode accessing registers and memory uses the privilege level derived in Table 1. Attempts from Debug Mode to access state that requires a privilege level above the **debug access privilege** will fail and set `abstractcs`.CMDERR to 3.

Table 1. 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

3.1.3. Maximum Allowed Resume Privilege Mode

The maximum privilege level that can be configured in `dcsr`.PRV and `dcsr`.V is determined in Section 3.1.3. 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 2. 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

3.1.4. External Debug Security Control Signals/Fields and CSRs

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

Table 3. 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

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

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

element in each hart, named `mdbgen`, is introduced to control the debuggability of M-mode.

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.

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 dependent 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 as depicted in Figure 1. 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 Program Buffer execution 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 to allow it to save/restore trigger context on behalf of a supervisor debugger. Otherwise, a trigger could serve as a side-channel to debug-disallowed supervisor domains. The trigger may raise a breakpoint exception in a supervisor domain where debugging is disallowed. This could allow the external debugger to indirectly observe the state from the debug-disallowed supervisor domain (PC, data address, etc.) and may even result in a Denial of Service (DoS). By making DMODE M-mode accessible when `mdbgen` is 0, such an attack can be mitigated by having M-mode firmware switch the trigger context at the supervisor domain boundary. Note that it remains only accessible to Debug Mode when `mdbgen` is 1.

If the hart is running in a debug-allowed privilege 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 the controlling entity 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 optional Smsdedbg extension is provided to control S/HS-mode external debug operations when M-mode external debug is disallowed. It introduces the `SDEDBGALW` field (bit 7) in CSR `msdcfg`, and it controls S/HS-mode external debug operations.

The `SDEDBGALW` field only has effect when `mdbgen` is 0; otherwise, debug control is dominated by the Smmdedbg extension.

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 Program Buffer execution operate with S/HS-mode privilege.
- The debug operations are allowed when the hart executes in S/HS-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` in S/HS-mode.
- Triggers are delegated to S/HS-mode as specified in Section 3.1.6.2, when Sdtrig is implemented.
- The **effective debug access privilege** of loads and stores in Debug Mode may be modified as described in Section 3.1.6.3.
- 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, S/HS-mode restrictions include:

- 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

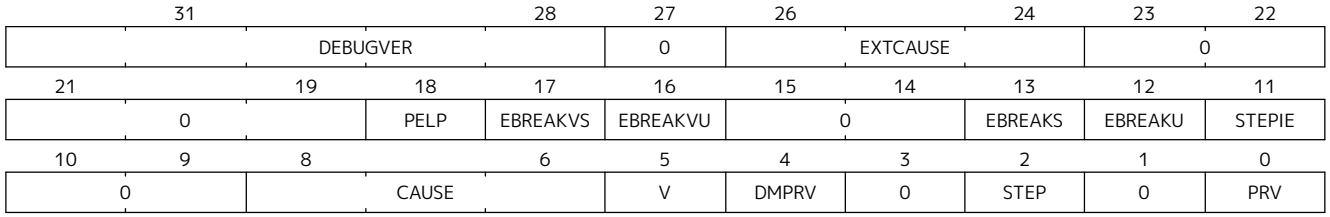
When SDEDBGALW is 1, the sdcsr and sdpc registers provide S/HS-mode read/write access to the dcsr and dpc registers respectively. However, sdcsr does not expose access to the MPRVEN field; instead, it repurposes the MPRVEN bit position with a DMPRV field to modify the **effective debug access privilege**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

Table 4. Allocated addresses for S/HS-mode shadow of Debug Mode CSR

Number	Name	Description
0xaaa	sdcsr	S/HS-mode debug control and status register.
0xaaa	sdpc	S/HS-mode debug program counter.

The sdcsr register exposes a subset of dcsr, formatted as shown in Register 1, 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 1: 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 sdcsr layout to avoid waste of fields.

Table 5. 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 3.1.6.3 and Section 3.1.7.2.	WARL	0

Extension of Sdtrig CSR

The Smtdeleg/Sstcfg [4] extensions define the process for delegating triggers to modes with lower privilege than M-mode. If Sdtrig is supported, the Sdsec requires both extensions to securely delegate Sdtrig triggers to the S/HS-mode.

When M-mode enables debugging for the S/HS-mode, it can optionally delegate the triggers to the S/HS-mode, allowing an external debugger with S/HS-mode privilege to configure these triggers.

Debug Access Privilege to memory in Smsdedbg

The `sdcsr`.DMPRV has 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 1; when `sdcsr`.DMPRV=1, the **effective debug access privilege** of loads and stores in Debug Mode is represented by:

- `sstatus`.SPP or,
- `hstatus`.SPVP 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 optional Smvsdedbg extension is provided to control VS-mode external debug operations when S/HS-mode external debug is disallowed. It introduces the `VSEDBGALW` field (bit TBD) in CSR `msdcfg`, and it controls VS-mode external debug operations.

The `VSEDBGALW` field only has effect when both `mdbgen` and `SDEDBGALW` are 0; otherwise, debug control is dominated by the Smmdedbg or Smsdedbg extensions.

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 Program Buffer execution operate with VS-mode privilege.
- The debug operations are allowed when the hart executes in VS-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 Section 3.1.6.1 in 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 lower privilege mode, VS-mode restrictions include:

- 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 privilege 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 privilege mode (i.e., SRET with `vsstatus`.SPP=0). The hart does not re-enter Debug Mode if the SRET instruction returns to a debug-disallowed privilege mode (i.e., SRET with `vsstatus`.SPP=1, `VSEDBGALW`=0).

`sdcsr` and `sdpc` in VS-mode

When `VSEDBGALW` is 1, the `sdcsr` and `sdpc` Section 3.1.6.1 are accessible with virtual supervisor 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 (reads return 0). Similar to `sdcsr` access when `SDEDBGALW` is 1, `sdcsr`.DMPRV modifies the **effective debug access privilege** in VS-mode. When external debug is disallowed in VS-mode, the configuration visible in `sdcsr` will be ignored as if it were 0 while the hart runs in VS-mode.

Redirected access to `dcsr`.EBREAKVS and `dcsr`.EBREAKVU unifies the configuration for both S/HS-mode and VS-mode. 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 1; 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 optional Smudedbg extension is provided to control U-mode and VU-mode external debug operations when S/HS-mode and VS-mode external debug are disallowed. It introduces the `USEDBGALW` field (bit TBD) in CSR `msdcfg`, and it controls U-mode and VU-mode external debug operations.

The `USEDBGALW` field only has effect when all implemented higher privilege mode debug controls are 0 (i.e., `mdbgen`=0 and any implemented `SDEDBGALW`/`VSEDBGALW`=0); otherwise, debug control is dominated by higher privilege debug extensions. The specific conditions depend on which optional debug extensions are implemented as shown in Table 10.

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 Program Buffer execution 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` 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 and the hart is running in U-mode (or VU-mode when virtualization mode is 1), 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 U-mode or VU-mode accessible CSRs (e.g.,`udcsr`, `udpc`) will fail and `abstractcs`.CMDERR is set to 3 (exception).
- The U-mode or VU-mode visible configuration in `udcsr` will be ignored as if it were 0.

`udcsr` and `udpc`

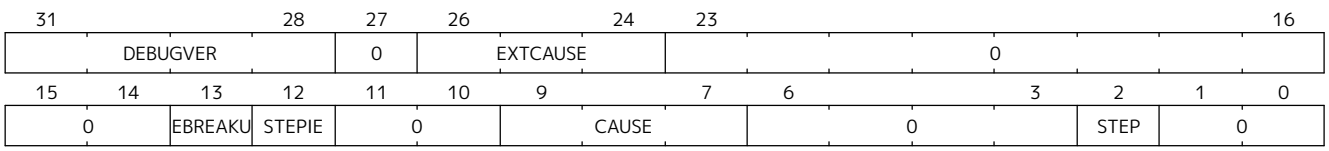
The `udcsr` and `udpc` registers provide U-mode or VU-mode read/write access to the `dcsr` and `dpc` registers respectively. The `udcsr` exposes a subset of `dcsr` and the accessible fields are listed in Register 2. The read/write access to the `udcsr`.EBREAKU field is redirected to `dcsr`.EBREAKVU when 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 6. 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 2, while the `udpc` register provides full access to `dpc`.



Register 2: 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.

The availability of trace output is controlled through signals defined in the hart-trace interface (HTI) [6]. The logical `sec_inhibit` *signal can be converted to the canonical trace interface signals by implementation.*

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

Table 7. Trace Control and CSRs

Extension	Trace Control	New CSRs	Target Modes
Smmdetrc	mtrcen	None	M-mode
Smsdetrc	msdcfg.SDETRCALW	None	S/HS-mode
Smvsdetrc	msdcfg.VSETRCALW	None	VS-mode
Smudetrc	msdcfg.USETRCALW	None	U-Mode or VU-mode

3.2.1. 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.

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 `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.

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

The optional Smsdetrc extension is provided to control S/HS-mode trace operations when M-mode trace is disallowed. It introduces the `SDETRCALW` field (bit 8) in CSR `msdcfg`, and it controls S/HS-mode trace output.

The `SDETRCALW` field only has effect when `mtrcen` is 0; otherwise, trace control is dominated by the Smmdetrc extension.

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.

3.2.3. VS-mode Trace Security Extension (Smvsdetrc)

The optional Smvsdetrc extension is provided to control VS-mode trace operations when S/HS-mode trace is disallowed. It introduces the `VSETRCALW` field (bit TBD) in CSR `msdcfg`, and it controls VS-mode trace output.

The `VSETRCALW` field only has effect when both `mtrcen` and `SDETRCALW` are 0; otherwise, trace control is dominated by the Smmdetrc or Smsdetrc extensions.

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.

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

The optional Smudetrc extension is provided to control U-mode and VU-mode trace operations when S/HS-mode and VS-mode trace are disallowed. It introduces the `USETRCALW` field (bit TBD) in CSR `msdcfg`, and it controls U-mode and VU-mode trace operations.

The `USETRCALW` field only has effect when all implemented higher privilege mode trace controls are 0 (i.e., `mtrcen`=0 and any implemented `SDETRCALW`/`VSETRCALW`=0); otherwise, trace control is dominated by higher privilege trace extensions. The specific conditions depend on which optional trace extensions are implemented as shown in Table 11.

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 the 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 8. 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 [sdsecextdbg]. According to *The RISC-V Debug Specification* [1], a halt request must be responded to within one second. However, this constraint must be removed as the request might be pending due to situations where 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 is disallowed to be debugged, 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 NDMRESET 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 is allowed to be debugged; 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 [sdsecextdbg]. 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 **effective debug access privilege**, as defined in Section 3.1.2.

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 [7], WorldGuard [8]. 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 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 9.

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 8. 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 9. 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 diagram illustrates the reference implementation of security control for the debug and trace, respectively.

A.1. External Debug Security Control

As outlined in the specification, the dedicated debug security policy for a hart is enforced by platform state `nsecdbg`, hart state `mdbgen`, and the SDEDBGALW field inside the `msdcfg` CSR. Both the `nsecdbg` and `mdbgen` states can be accommodated in MMIO outside the harts, such as in the Debug Module registers, or implemented as fuses.

The security control logic validates all debug requests and triggers (with ACTION=1) matching or firing based on `nsecdbg`, `mdbgen`, and SDEDBGALW against the privilege level of the hart. Debug requests that fail validation will either be dropped or kept pending. Additionally, the platform-specific external trigger inputs must obey platform constraints, which must be carefully handled by the platform implementation.

When `nsecdbg` is set to 0, the validation process involves two actors, which may lead to a potential Time-of-Check Time-of-Use (TOCTOU) issue. To mitigate this, the implementation must ensure that both the validation and execution of debug requests occur under the same privilege level and the same debug security policy. Failing to do so may allow debug requests to bypass security controls.

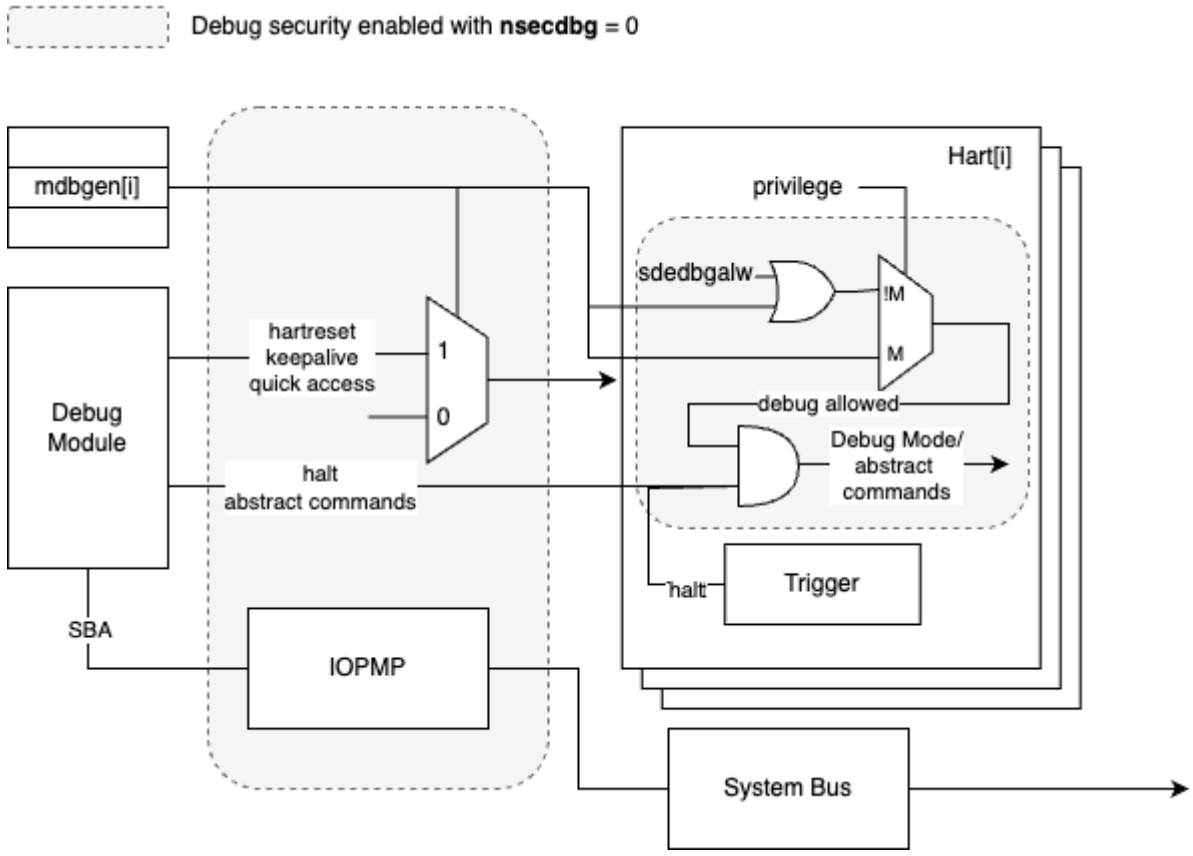


Figure 1. The debug security control

Application-level debugging is primarily accomplished through self-hosted debugging, allowing the management of debug policies by supervisor domains. As a result, user-level debugging management is not addressed within this extension.

The tables below show valid implementation combinations for different architectures for external debug:

Table 10. 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

A.2. Trace Security Control

Similar to debug security, trace is controlled by platform state `nsecdbg`, hart state `mtrcen`, and SDETRCALW in CSR `msdcfg` for each hart. The sec_inhibit sideband signal indicates the availability of trace to the trace encoder.

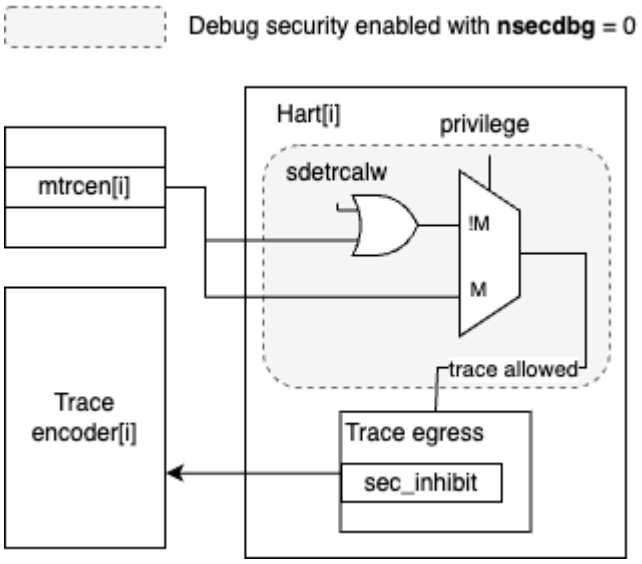


Figure 2. The trace security control

The tables below show valid implementation combinations for different architectures for trace:

Table 11. 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: Debug Policy Management

TBD: Elaborate on how to manage debug policies for different privilege modes by software.

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 the memory and CSR. 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 Trigger Delegation Specification.” [Online]. Available: github.com/riscv/ft-trigger-delegation.

[5] “RISC-V Supervisor Domains Access Protection.” [Online]. Available: github.com/riscv/riscv-smmtt.

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

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

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