RISC-V Debug Mode concept modification proposal

Rev 0.1, 2018-03-29

Syntacore <debug@syntacore.com>

Revision history

Version	Date	Description
0.1	2018-03-29	Initial revision

Introduction

The document contains Syntacore's proposal on reorganization of the Debug Mode, a key concept in RISC-V Debug Specification. This includes modification of the Debug Mode concept, extending the Mode's set of operational parameters and states, and sizable changes in the Debug CSRs.

This proposal references RISC-V Debug Specification draft 0.13, available here:

https://github.com/riscv/riscv-debug-spec

The source text for this proposal is available here:

https://github.com/syntacore/riscv-dbg-proposal

1. Debug Mode

1.1. Updated Debug Mode concept

Currently D-MODE is a special processor mode with behavior significantly different from behavior in normal modes. And that behavior is fixed and cannot be changed. We propose to generalize Debug Mode and treat it as a special mode with set of programmable run control parameters regulating all relevant aspects of HART behavior: PC advancement, exception enabling, events of redirection to D-MODE etc.

Main proposed changes in the Debug Mode concept:

1. **Unambiguous Debug Mode state machine.** Firstly, it is necessary to define a state machine associated with actions around Debug Mode and transitions between it and normal operational modes (U/S/M-modes). That gives us explicit definition of relevant debug states, conditions of transitions between them and permitted operations in each of them. Such a formal description makes specification of other debug aspects unambiguous and more precise and is included as part of this proposal.
2. **Debug Mode Run Control parameters.** Adding in the Debug Module register address space two sets of Run Control parameters: the first set - to define execution parameters in D-MODE, and second - for the execution in the normal mode after resuming from D-MODE. These Run Control Parameters can be defined, for example, as potential "departures" from the standard behavior in the corresponding mode. Then, implementation of those bits could be made optional for simpler implementations, and the case when all the bits are not implemented should be very close to the current Debug Mode behavior. Register with those Run Control Parameters can be added to the HART's Debug CSRs.
3. **Debug Mode redirection.** To introduce Debug Mode redirection mechanism, which should allow entering Debug Mode upon various events, including exceptions and interrupts. The redirection should be controllable on the per-privilege mode basis. Control status registers for this mechanism should be placed in the Debug CSR address space.
4. **Special Debug CSRs for exceptions processing.** The following exception-processing Debug CSRs are added: DEPC, DCAUSE, DTVAL. That provides important additional facilities:
    - Allows processing of exceptions raised during Program Buffer instructions execution (while HART is in Debug Mode)
    - Allows to introduce in exception processing model additional intermediate state - when exception is raised but not committed to architectural state, and on that basis to extend exception handling capabilities
5. **Debug Program Buffer PC register.** Debug Program Buffer PC register (DPPC CSR) is added for dealing with PC values during Program Buffer instructions execution. That helps in saving architectural PC value intact (inside DPC) over Debug Mode execution series.
6. **Debug Mode context saving.** Two CSRs (DSAVE0/1) are added to accommodate architectural context saving over a Debug Mode session. Those registers are expected to be used as a backup storage for 1-2 GPRs used as intermediate registers in Program Buffer procedures. That addition together with DPPC, DPC registers and PRV/DPRV bits in DRUNCONTROL allows storing of HART-

specific context of typical Debug Mode session inside the HART, and makes debug facilities more robust to debug session breaks.

1.2. Hart Debug States

The proposed set of debug states and their interdependencies are shown in Figure 1.

From the debug operations prospective HART has the following debug states (or macro-states, to be precise):

- RESET: state the HART has when its hardwire reset signal is activated
- RUN: corresponds to the normal instruction execution operational mode with U/S/M privileges
- HALTED: this is the state the HART has after WFI instruction execution
- D_HALTED (Debug Halted): this is the state the HART has upon the entry into the Debug Mode
- D_RUN (Debug Run): corresponds to the state when HART executes instructions from Program Buffer while it is in Debug Mode

RUN and HALTED states correspond to the Non-Debug (or Normal) Mode, whereas D_HALTED and D_RUN states comprise Debug Mode.

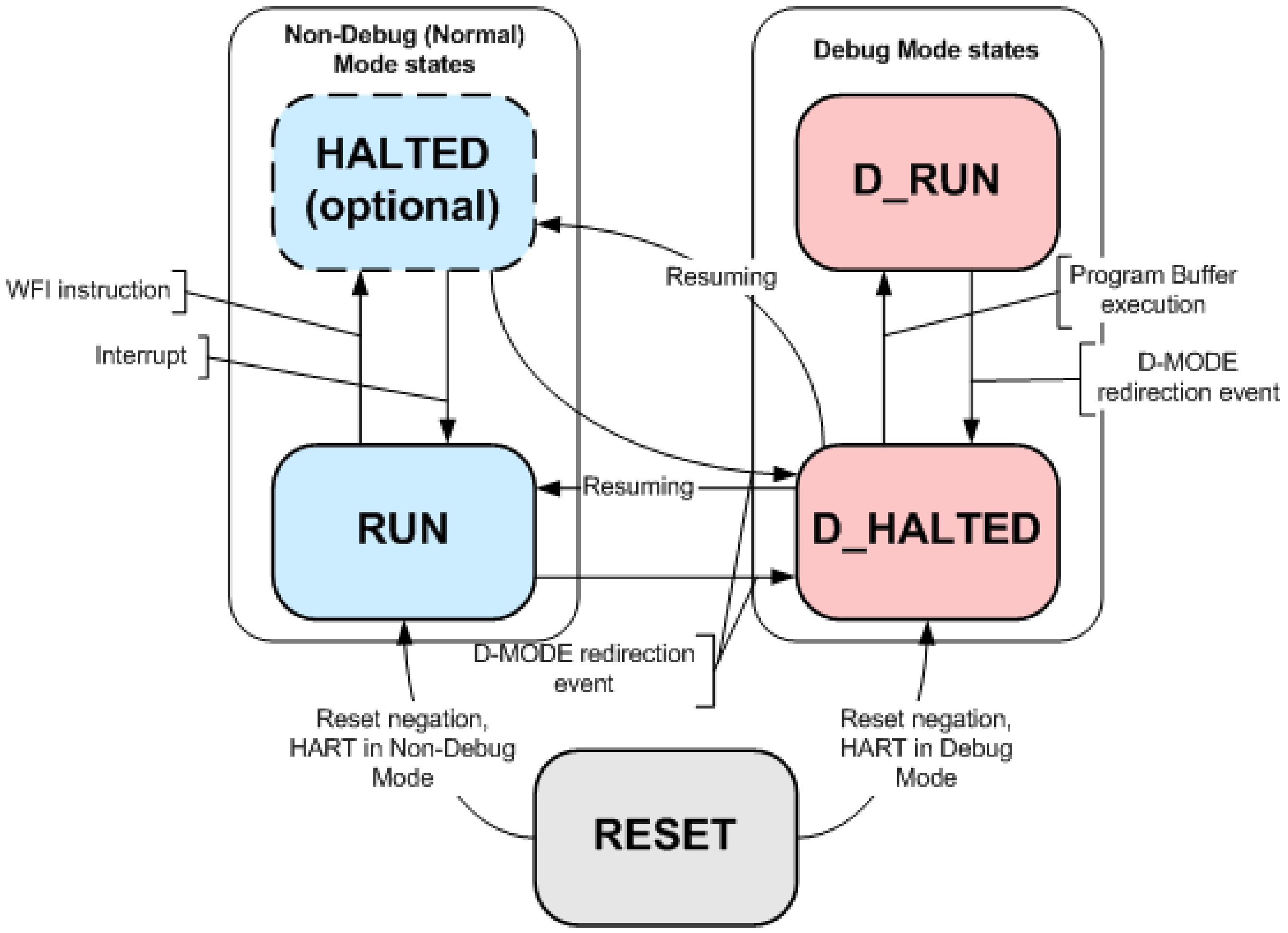


Figure 1. Debug States of HART

1.2.1. RESET

HART transits to RESET state from any other state upon assertion of its hardwire reset signal.

All HART architectural registers (GPRs, FPRs, CSRs) changes their values to the reset-specific ones as defined in the RISC-V ISA Spec.

In that state all HART-specific debug parameters (called Debug Context), including Debug CSRs, stay intact, as they are stored in the Debug Interface module independent of the HART's reset. Debug Interface is described in the section "Interaction between Debug Module and HART".

After reset signal de-assertion HART may transit to one of the following states:

- RUN, if there is no debug halt request from the HART's Debug Interface.
- D_HALTED, if HART's Debug Interface activates debug halt request. That may be the case when at least one of the following is true: 1) HART was in Debug Mode before reset assertion; or 2) Debug Module asserted halt request for the given HART.

1.2.2. RUN

RUN is the normal HART instruction execution operational mode with U/S/M privileges. Its behavior is fully defined by the RISC-V ISA spec and depends on HART's architectural state (set of all HARTs GPRs, FPRs and CSRs).

HART may transit to the RUN state from:

- RESET state - after hardware reset signal de-asserion, if the HART is in Non-Debug Mode
- D_HALTED state - as a result of resuming from Debug Mode
- HALTED (optional) - after an interrupt activation

HART may leave the RUN state to:

- D_HALTED state - as a result of redirection to Debug Mode upon external (debug halt request) or internal (single step, EBREAK, hardware breakpoint, exception redirection etc) events.
- HALTED state (optional) - after WFI instruction execution.

1.2.3. HALTED (optional)

HALTED is the state the HART has after WFI instruction execution. Upon entry, HART is halted until the next active interrupt. That state could be considered as optional, or to be treated as a sub-state of RUN.

1.2.4. D_HALTED

D_HALTED is the state the HART has after entry to Debug Mode. HART is halted, and operational context is changed to the one of Debug Mode, and HART's behavior is defined by Debug Spec.

HART may transit to the D_HALTED state from:

- RESET state - after hardware reset signal de-asserion, if the HART is in Debug Mode
- RUN state - as a result of redirection to Debug Mode upon external (debug halt request) or internal (single step, EBREAK, hardware breakpoint, exception redirection etc) events
- HALTED (optional) - as a result of redirection to Debug Mode
- D_RUN state - as a result of redirection back to D_HALTED state after Program Buffer program completion or unexpected exception raising during that program execution

HART may leave the D_HALTED state to:

- RUN state - as a result of resuming from Debug Mode
- HALTED state (optional) - as a result of resuming from Debug Mode with DRUNCONTROL.wfi bit set
- D_RUN state - when Program Buffer execution is started

1.2.5. D_RUN

D_RUN is the state when HART executes instructions from Program Buffer while it is in Debug Mode. HART's behavior is defined by Debug Spec.

HART transits to the D_RUN state from D_HALTED state when Program Buffer execution is started.

HART leaves the D_RUN state toward D_HALTED state as a result of redirection back after Program Buffer program completion or unexpected exception raising during that program execution.

1.3. Interaction between Debug Module and HART

All interactions between Debug Module (DM) and HART are carried out via HART Debug Interface Module (HDIM). HDIM has a set of its own states controlling debug behavior of HART:

- Debug CSRs - these are considered as a part of HDIM
- some set of micro-architectural states (registers possibly not exposed externally) supporting Debug States and transitions between them

That set of HDIM internal states is called HART Debug Context.

HDIM formally can be considered as a part of a HART, but its reset is independent from HART's reset. HDIM must be reset together with the Debug Module.

1.4. Run Control Parameters

Central point in the proposed new Debug Mode concept is the change from the fixed, very specific run control setup for execution in Debug Mode context, to an orthogonal set of programmable run control parameters regulating all debug relevant aspects of the HART behavior.

Two sets of such run control parameters are required:

- Debug Run Control Parameters (DRCPs) - parameters regulating HART behavior while it is in

Debug Mode, during execution of Program Buffer instructions (in the D_RUN state)

- Run Control Parameters (RCPs) - parameters regulating HART behavior after resuming from Debug Mode, while it is in the RUN state

Basically, these parameters define potential "departures" from the standard behavior in the corresponding mode. Standard behavior of HART for Debug and Non-Debug Modes can be different. Then, implementation of those parameters could be made optional for simpler implementations, and the case when all the bits are not implemented should be very close to the current Debug Mode behavior.

All RUN Control Parameters are in the Debug Run Control CSR (DRUNCONTROL) (refer to section 1.6). DRCPs and RCPs are located in the upper and lower halves of the register, respectively.

RCPs are used after resuming from Debug Mode, when HART is in the RUN state. That group includes the following parameters:

- Privilege Level - the field behaves like the PRV field in the DCSR of the Debug Spec 0.13
- Single Step option - enforces redirection to D-MODE after single instruction execution
- Interrupt Disable - prohibits interrupts in RUN state
- Redirecting Exception Disable - prohibits commit of the exception-related changes into architectural state upon redirection to D-MODE caused by that exception. The parameter influences processing of all exceptions but Breakpoint exception
- Redirecting EBREAK Enable - allows commit of changes into architectural state related to the Breakpoint exception (due to both EBREAK execution and hardware breakpoint from Trigger Module). The parameter inverts default redirection to Debug Mode behavior upon Breakpoint exception (which is to discard architectural changes)
- Exception Commit - control flag, forces HART to commit exception data from Debug CSRs (DCAUSE, DEPC, DTVAL) to the corresponding architectural CSRs after resuming from D-MODE. The option is described in the section "Exception /Interrupt Handling"
- Wait For Interrupt - an option allowing to transit to the HALTED state after resume, i.e., the state identical to the one after WFI instruction execution
- Trigger Module Disable - option prohibiting Trigger Module actions in the RUN state.
- Trace Disable - option reserved for future use with the Trace Module

DRCPs are used when HART is in the D_RUN state. That group includes the following parameters:

- Debug Mode Privilege Level - the field is used to control privilege level of the Program Buffer’s instructions
- Debug Single Step - allows debugging of the Program Buffer’s code
- Debug Interrupt Enable - allows interrupts in the D_RUN state; could be used for interrupt processing by the Program Buffer procedures
- Debug Redirecting Exception Enable - allows commit of the exception-related changes into architectural state upon redirection to the D_HALTED state caused by that exception. The parameter influences processing of all exceptions but Breakpoint exception

- Debug Redirecting EBREAK Enable - allows commit of changes into architectural state related to the Breakpoint exception
- Debug Exception Commit - control flag, forces HART to commit exception data from Debug CSRs (DCAUSE, DEPC, DTVAL) to corresponding architectural CSRs after transition to D_RUN state. The option is described in the section "Exception /Interrupt Handling"
- Debug Wait For Interrupt - allows to imitate Wait-For-Interrupt state during instructions execution in the D_RUN state
- Debug Trigger Module Enable - allows Trigger Module actions in the D_RUN state.
- Debug Trace Enable - option reserved for future use with the Trace Module; it should allow tracing of Program Buffer's instructions execution
- Debug PC Altering Enable - allows changing of Program Counter during the Program Buffer execution. By default, the PC altering is disabled
- Debug Privilege Level Altering Instructions Enable - allows proper execution of the instructions, changing privilege level; by default such instructions are prohibited and cause Illegal Instruction exception
- Debug Counter Enable - allows increment of HART-local counters in the D_RUN state

All DRCPs except Debug Mode Privilege Level field are cleared after transition from the RUN or the HALTED state to the D_HALTED state, ensuring default behavior after redirection to Debug Mode. After transition from the RUN or the HALTED state to the D_HALTED state the Privilege Level field contains the privilege level value the HART had in the RUN state just before that transition. That means the default privilege level is the same as it was in the program just interrupted. Transition from the D-RUN to the D_HALTED state itself does not change parameter values, however, the privilege level might be changed during Program Buffer instructions execution, if that is allowed by DRUNCONTROL.dprv_inst_enbl bit.

1.5. Exception/Interrupt Handling

Another proposed feature of the Debug Mode is capability of flexible exception/interrupt handling in Debug Mode for debugging or testing purposes. This feature is enabled by a combination of the already introduced modifications in the Debug Mode concept:

- Introducing of D-MODE redirections due to exceptions/interrupts
- Treating of D-MODE redirection as a sort of exception, with registering its details in Debug CSRs (DCAUSE, DEPC, DTVAL)
- Adding of the Run Control Parameters, in particular, Redirecting Exception Disable, Redirecting EBREAK Enable and their D_RUN counterparts

The state diagram of debug exception handling is shown in Figure 2.

Conventional exception handling by software in normal, Non-Debug Mode can be described as a sequence of the following steps:

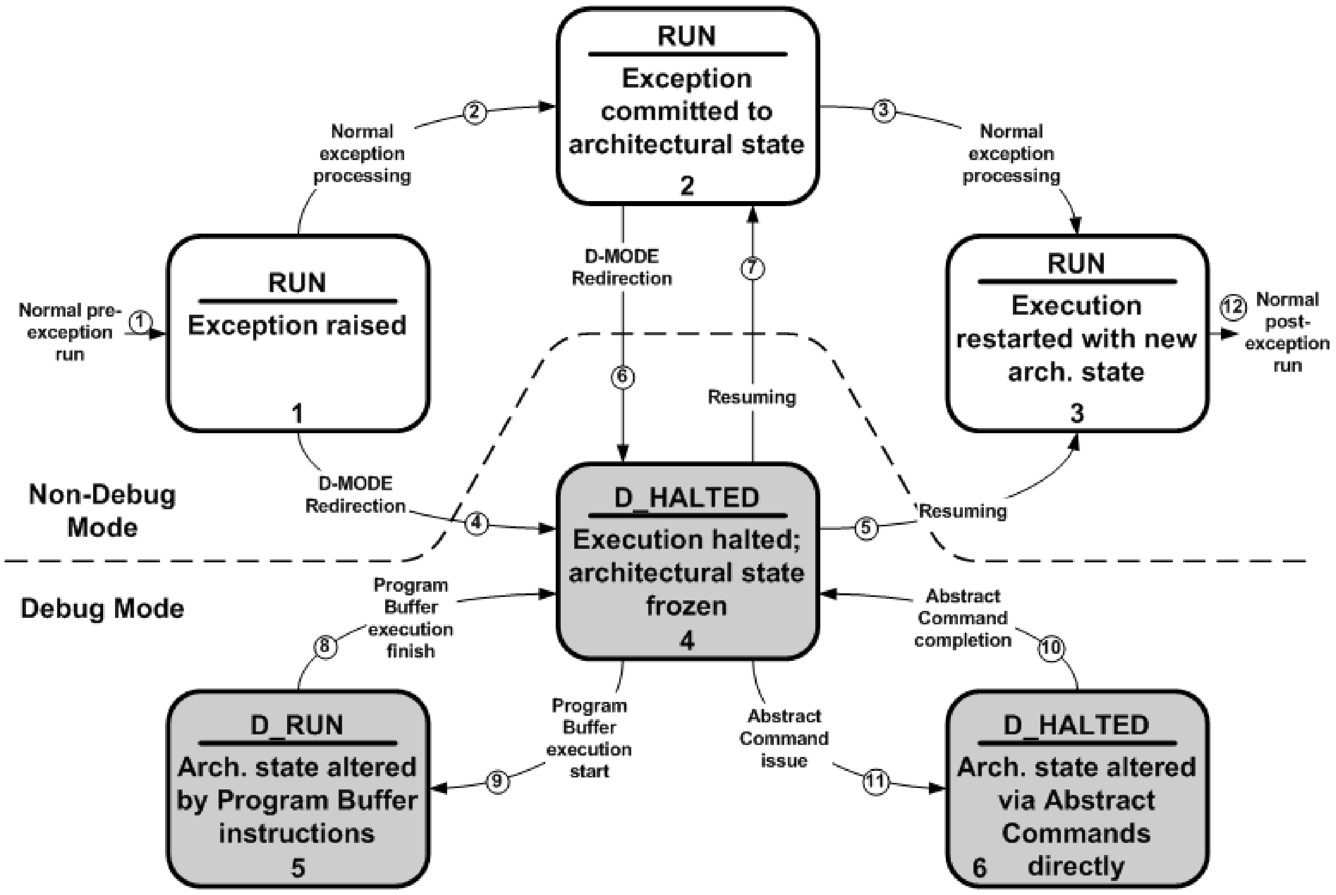
- Exception assertion (sub-state 1 in the diagram): it is a situation when an exception is detected, HART's pipeline is halted and flushed, but exception data is not committed to the HART's architectural state yet

- Exception commit (sub-state 2): the fact of the exception and all relevant exception data are committed to architectural state; HART's architectural state is also modified for a jump to a software exception handler vector
- Execution restarts with the new architectural state (sub-state 3): pipeline performs jump to a software exception handler vector, and that restarts execution from the new point and with the new architectural context

Capability of D-MODE redirection upon exception and Run Control Parameters extend this picture with two variants of D-MODE redirection (transitions 4 and 6), and two variants of resuming from D-MODE (transitions 5 and 7).

Transition 4: D-MODE redirection w/o exception commit

It takes place if DRUNCONTROL.re_dsbl = 1 (for all exceptions except Breakpoint) or DRUNCONTROL.re_break_en = 0 (for Breakpoint exception). Exception data is registered only in the exception-related Debug CSRs (DCAUSE, DEPC, DTVAL), the architectural CSRs do not contain information about the exception which triggered the redirection.



RUN
Exception committed to architectural state
2
Normal exception processing
2
3
Normal exception processing
RUN
Exception raised
1
Normal pre-exception run
1
D-MODE Redirection
6
7
Resuming
RUN
Execution restarted with new arch. state
3
12
Normal post-exception run
Non-Debug Mode
D-MODE Redirection
4
D_HALTED
Execution halted; architectural state frozen
4
5
Resuming
Debug Mode
Program Buffer execution finish
8
Abstract Command completion
10
D_RUN
Arch. state altered by Program Buffer instructions
5
Program Buffer execution start
9
Abstract Command issue
11
D_HALTED
Arch. state altered via Abstract Commands directly
6

Figure 2. Debug exception handling state diagram

Transition 6: D-MODE redirection with exception commitment

It takes place if DRUNCONTROL.re_dsbl = 0 (for all exceptions except Breakpoint) or DRUNCONTROL.re_break_en = 1 (for Breakpoint exception). Exception data is registered both in the exception-related Debug CSRs (DCAUSE, DEPC, DTVAL) and in the architectural CSRs.

Transition 5: simple resume from D-MODE (to the RUN state)

If debugger requests traditional resume from D-MODE (the one described in the Debug Spec currently), the HART transits from sub-state 4 to sub-state 3 via ark 5. In that case, HART mainly inherits the architectural state it had just before that transition (with an exception of the privilege level). During that transition a set of Run Control Parameters (RCPs) is also applied including the privilege level for sub-state 3 and the following execution. RCPs are taken from the lower half of DRUNCONTROL register.

The transition could be used in the following cases:

- if HART is redirected to sub-state 4 via transition 4 (w/o exception commit), and debugger wants to repeat execution of the instruction triggering the exception
- if HART is redirected via transition 6 (with exception commit), and debugger wants to continue exception processing in software
- independently of redirection method, debugger on its own made some modifications in the architectural state (i.e. with debugging purposes) while being in D-MODE, and wants to restart execution from the new point

Transition 7: resuming from D-MODE with exception data commit

This transition is similar to the transition 5 with one difference: exception-relevant data are copied from Debug CSRs (DCAUSE, DEPC, DTVAL) to the corresponding architectural CSRs, as well as exception indication. So, in that case HART will continue to process the exception (interrupted by redirection or even composed artificially in D-MODE by debugger).

To initiate resume via transition 7, bit DRUNCONTROL.e_commit must be set.

Architectural state read and modification

While HART is in D-MODE, its architectural state (GPRs, FPRs, CSRs) can be read and modified in two ways:

- directly by Abstract Commands (if that is implemented for the corresponding registers). In the diagram this is reflected by sub-state 6 and transitions 10 and 11.
- by the Program Buffer instructions execution (it’s done indirectly by Abstract Commands), if the Program Buffer is implemented. In the diagram that corresponds to sub-state 5 and transitions 8 and 9.

1.6. Core Debug CSRs

1.6.1. Register map

Table 1. Core Debug CSRs. Memory Map

Address	Short Name	Name/Description
0x7b0	DCSR	Debug Control Status Register

Address	Short Name	Name/Description
0x7b1	DPC	Debug Program Counter Register
0x7b2	DSCRATCH0	Debug Scratch Register 0
0x7b3	DSCRATCH1	Debug Scratch Register 1
0x7b4	DEPC	Debug Exception Program Counter Register
0x7b5	DCAUSE	Debug Cause Register
0x7b6	DTVAL	Debug Trap Value Register
0x7b7	DRUNCONTROL	Debug Run Control Register
0x7b8	DEREDIREN_M	Debug M-MODE Exception Redirection Enable Register
0x7b9	DEREDIREN_S	Debug S-MODE Exception Redirection Enable Register
0x7ba	DEREDIREN_U	Debug U-MODE Exception Redirection Enable Register
0x7bb	DIREDIREN	Debug Interrupt Redirection Enable Register
0x7bc	DPPC	Debug Program Buffer PC Register
0x7bd	Reserved	Reserved for future use
0x7be	DSAVE0	Debug Save Register 0
0x7bf	DSAVE1	Debug Save Register 1

1.6.2. DCSR, Debug Control Status Register (0x7b0)

Table 2. Debug Control Status Register

Bit(s)	Name	Access	Reset	Description
1:0	state	RO	0x0	Debug State. The field indicates the current debug state. Encoding: • 0b00 : RESET state; • 0b01 : RUN or HALTED/WFI state; • 0b10 : D_RUN state; • 0b11 : D_HALTED state.
3:2	pstate	RO	0x0	Previous Debug State. The field indicates the previous debug state. Encoding: • 0b00 : RESET state; • 0b01 : RUN or HALTED/WFI state; • 0b10 : D_RUN state; • 0b11 : D_HALTED state.

Bit(s)	Name	Access	Reset	Description
7:4	cause	RO	0x0	Debug Mode Entrance Cause. Explains why Debug Mode was entered. When there are multiple reasons to enter Debug Mode in a single cycle, the cause with the highest priority is the one written. Encoding: • 0: reserved as the field's reset value • 1: EBREAK instruction execution (priority 3) • 2: breakpoint in Trigger Module, direct redirection (priority 4) • 3: breakpoint in Trigger Module, redirection via exception (priority 4) • 4: Debug Mode redirection request from Debug Module Interface (priority 2) • 5: single step (priority 1) • 6: exception/interrupt (excluding breakpoint exception, priority for exception: 3 for interrupt: 1) • 7: NMI (priority 1) • 8: reset assertion (priority 5) • 9: reset negation (priority 1) • others: reserved
11:8	Reserved	RO	0x0	Reserved for future use.
12	stopcount	RW	0	Stop Counters. The value is applied in the D_HALTED debug state. Encoding: • 0: Increment counters as usual • 1: Don't increment any hart-local counters while in the D_HALTED state
15:13	Reserved	RO	0x0	Reserved for future use

Bit(s)	Name	Access	Reset	Description
19:16	gredir_en	RW	0x0	General Redirection Enable. The field contains bits determining those Debug Mode redirection events which are independent from privilege levels and/or having global influence over HART. If the corresponding bit is set and the corresponding event takes place, then HART transits to Debug Mode (to D_HALTED state). Bits: • 0: Reset assertion • 1: Reset negation (de-assertion) • 2: NMI • others: reserved
26:20	Reserved	RO	0x0	Reserved for future use
27	nmi_status	RO	0	NMI Status. The bit reflects status of NMI. It is set, if NMI is pending
31:28	xdebugver	RO	IMPL-DEF	External Debug Facility Version. Has the same meaning as per Debug Spec 0.13. Encoding: • 0: There is no external debug support • 4: External debug support exists as it is described in RISC-V Debug Specification 0.13 • 5: External debug support exists as it is described in this proposal • 15: There is external debug support, but it does not conform to any available version of RISC-V Debug Specification • others: reserved

1.6.3. DPC, Debug Program Counter Register (0x7b1)

The behavior of DPC register is close to the one described in the Debug Spec 0.13.

Upon entry to the D_HALTED state from the RUN state, DPC is updated with the virtual address of the next instruction to be executed. When resuming (transition to the RUN state), the HART's PC is updated to the virtual address stored in DPC. A debugger may write DPC to change where the HART resumes.

1.6.4. DSCRATCH0/1, Debug Scratch Register 0/1 (0x7b2/0x7b3)

The register is used for data exchange between HART and Debug Module during Program Buffer's instructions execution.

1.6.5. DEPC, Debug Exception Program Counter Register (0x7b4)

Upon entry to the D_HALTED state (both from RUN and from D_RUN states), DEPC behaves as the corresponding view of MEPC (UEPC, SEPC or MEPC itself), depending on the privilege level in the previous state.

1.6.6. DCAUSE, Debug Cause Register (0x7b5)

Upon entry to the D_HALTED state (both from RUN and from D_RUN states), DCAUSE behaves as the corresponding view of MCAUSE (UCAUSE, SCAUSE or MCAUSE itself), depending on the privilege level in the previous state.

1.6.7. DTVAL, Debug Trap Value Register (0x7b6)

Upon entry to the D_HALTED state (both from RUN and from D_RUN states), DTVAL behaves as the corresponding view of MTVAL (UTVAL, STVAL or MTVAL itself) depending on the privilege level in the previous state.

1.6.8. DRUNCONTROL, Debug Run Control Register (0x7b7)

Table 3. Debug Run Control Register

Bit(s)	Name	Access	Reset	Description
1:0	prv	R/W	0x0	Privilege Level. The value is associated with execution in RUN state. It is applied only during transitions between RUN and D_HALTED states. After transition from the RUN to the D_HALTED state, the field contains the privilege level value the HART had in the RUN state just before that transition. During transition from the D_HALTED to the RUN state the field's value is applied as the new HART's privilege level.
2	step	R/W	0	Single Step. The value is applied during execution in the RUN state. If set, after transition from the D_HALTED to the RUN state the HART executes a single instruction and then redirects back to the D_HALTED state.

Bit(s)	Name	Access	Reset	Description
3	i_dsbl	R/W	0	Interrupt Disable. The value is applied during execution in the RUN state. If set, the HART’s interrupts are disabled after transition from the D_HALTED to the RUN state.
4	re_dsbl	R/W	0	Redirecting Exception Disable. Affects exception handling in the RUN state for all exceptions but the Breakpoint exception. Encoding: • 0: the HART commits exception-related data to its architectural state, when it transits to the D_HALTED state upon corresponding exception redirection. The exception details are also reflected in the Debug CSRs (DCAUSE, DEPC, DTVAL). • 1: the HART does not commit exception-related data to its architectural state, when it transits to the D_HALTED state upon corresponding exception redirection. The exception details are reflected only in the Debug CSRs (DCAUSE, DEPC, DTVAL).
5	re_break_en	R/W	0	Redirecting EBREAK Enable. The value affects EBREAK exception processing in the RUN state: • 0: the HART does not commit Breakpoint exception-related data to its architectural state, when it transits to the D_HALTED state upon the Breakpoint exception redirection (as there was no execution of EBREAK at all). The exception details are reflected only in the Debug CSRs (DCAUSE, DEPC, DTVAL). • 1: the HART commits Breakpoint exception-related data to its architectural state, when it transits to the D_HALTED state upon the Breakpoint exception redirection. The exception details are also reflected in the Debug CSRs (DCAUSE, DEPC, DTVAL). This mode could be used for debugging of a software Breakpoint exception handler.

Bit(s)	Name	Access	Reset	Description
6	e_commit	R/W	0	Exception Commit. The value is applied during execution in RUN state. Encoding: • 0: Exception relevant data is not transferred from Debug CSRs to HART's architectural state upon transition from the D_HALTED to the RUN state. • 1: Exception relevant data is transferred from Debug CSRs to HART's architectural state during transition from the D_HALTED to the RUN state. This include: ◦ Exception event indication to the HART ◦ Content of DCAUSE, DEPC, DTVAL is copied to proper M/S/U CSRs
7	wfi	R/W	0	Wait For Interrupt. The value is applied during execution in RUN state. Encoding: • 0: After transition from the D_HALTED to the RUN state the HART starts to execute instructions from address specified in DPC register • 1: After transition from the D_HALTED to the RUN state the HART transits to the same state as after WFI instruction execution
8	tm_dsbl	R/W	0	Trigger Module Disable. The value is applied during execution in RUN state. Encoding: • 0: After transition from the D_HALTED to the RUN state the HART's Trigger Module works normally • 1: After transition from the D_HALTED to the RUN state the HART's Trigger Module is disabled, i.e. Trigger Module does not track program execution and does not react on hardware breakpoints hits
9	trace_dsbl	R/W	0	Trace Disable. The field is reserved for interaction with the future processor trace facility. The value is applied during execution in the RUN state. Encoding: • 0: After transition from the D_HALTED to the RUN state tracing of the HART program flow works normally • 1: After transition from the D_HALTED to the RUN state tracing of the HART program flow is disabled

Bit(s)	Name	Access	Reset	Description
15:10	Reserved	RO	0x0	Reserved for future use
17:16	dprv	R/W	0x0	Debug Mode Privilege Level. The value is associated with execution in the D_RUN state. It is applied mainly during transitions between the D_HALTED and the D_RUN states. After transition from the RUN or the D_RUN to the D_HALTED state the field contains the privilege level value the HART had in the RUN/D_RUN state just before that transition. During transition from the D_HALTED to the D_RUN state the field's value is applied as the new HART's privilege level.
18	dstep	R/W	0	Debug Single Step. The value is applied during execution in the D_RUN state. If set, after transition from the D_HALTED to the D_RUN state the HART executes a single instruction and then redirects back to the D_HALTED state.
19	di_enbl	R/W	0	Debug Interrupt Enable. The value is applied during execution in the D_RUN state. If set, the HART's interrupts are enabled after transition from the D_HALTED to the D_RUN state.

Bit(s)	Name	Access	Reset	Description
20	dre_enbl	R/W	0	Debug Redirecting Exception Enable. The value is applied during execution in the D_RUN state. It influences handling of all exceptions but Breakpoint exception. Encoding: • 0: the HART does not commit exception-related data to its architectural state, when it transits to the D_HALTED state upon corresponding exception redirection occurred in the D_RUN state. However, the exception details are still reflected in the Debug CSRs (DCAUSE, DEPC, DTVAL). • 1: the HART commits exception-related data to its architectural state, when it transits to the D_HALTED state upon corresponding exception redirection. As usual, the exception details are also reflected in the Debug CSRs (DCAUSE, DEPC, DTVAL).
21	dre_break_en	R/W	0	Debug Redirecting EBREAK Enable. The value is applied during execution in the D_RUN state. Encoding: • 0: the HART does not commit Breakpoint exception-related data to its architectural state, when it transits to the D_HALTED state upon the Breakpoint exception redirection (as there was no execution of EBREAK at all). However, those details are still reflected in Debug CSRs (DCAUSE, DEPC, DTVAL). • 1: the HART commits Breakpoint exception-related data to its architectural state, when it transits to the D_HALTED state upon the Breakpoint exception redirection. The exception details are also reflected in the Debug CSRs (DCAUSE, DEPC, DTVAL).

Bit(s)	Name	Access	Reset	Description
22	de_commit	R/W	0	Debug Exception Commit. The value is applied during execution in the D_RUN state. Encoding: • 0: Exception-relevant data is not transferred from Debug CSRs to HART’s architectural state during transition from the D_HALTED to the D_RUN state. • 1: Exception-relevant data is transferred from Debug CSRs to HART’s architectural state during transition from the D_HALTED to the D_RUN state. This includes: ◦ Exception event indication to the HART ◦ Content of DCAUSE, DEPC, DTVAL is copied to proper M/S/U CSRs
23	dwfi	R/W	0	Debug Wait For Interrupt. The value is applied during execution in the D_RUN state. Encoding: • 0: After transition from the D_HALTED to the D_RUN state the HART starts executing instructions from the Program Buffer • 1: After transition from the D_HALTED to the RUN state the HART transits to the same state as after WFI instruction execution
24	dtm_enbl	R/W	0	Debug Trigger Module Enable. The value is applied during execution in the D_RUN state. Encoding: • 0: After transition from the D_HALTED to the D_RUN state the HART’s Trigger Module is disabled, i.e. Trigger Module does not track program execution and does not react on the hardware breakpoints hits • 1: After transition from the D_HALTED to the D_RUN state the HART’s Trigger Module works normally

Bit(s)	Name	Access	Reset	Description
25	dtrace_enbl	R/W	0	Debug Trace Disable. The field is reserved for interaction with the future processor trace facility. The value is applied during execution in the D_RUN state. Encoding: • 0: After transition from the D_HALTED to the D_RUN state tracing of the HART program flow is disabled • 1: After transition from the D_HALTED to the D_RUN state tracing of the HART program flow works normally
26	dpc_alt_enbl	R/W	0	Debug PC Altering Enable. The value is applied during execution in the D_RUN state. Encoding: • 0: PC altering is disabled. Program Counter is frozen • 1: PC altering is enabled. Program Counter changes normally during instructions execution from the Program Buffer
27	dprv_inst_en bl	R/W	0	Debug Privilege Level Altering Instructions Enable. The value is applied during execution in the D_RUN state. List of instructions includes: ECALL, MRET, SRET, and URET. The only exception is EBREAK (it always redirects HART to the D_HALTED state). Encoding: • 0: Instructions changing the privilege level are disabled. If HART executes such an instruction, the HART behaves like Illegal Instruction exception is raised with dre_enbl bit cleared. That means redirection to the D_HALTED state with correspondent Illegal Instruction exception details indication in DCAUSE, DEPC and DTVAL registers. • 1: Instructions changing the privilege level are enabled. If HART executes such an instruction, the HART behaves normally, as it is specified in RISC_V ISA spec.
28	dcount_enbl	R/W	0	Debug Counter Enable. The value is applied during execution in D_RUN state. Encoding: • 0: Don’t increment any hart-local counters • 1: Increment counters as usual

Bit(s)	Name	Access	Reset	Description
31:29	Reserved	RO	0x0	Reserved for future use

1.6.9. DEREDIREN_M, Debug M-MODE Exception Redirection Enable Register (0x7b8)

DEREDIREN_M has a bit position allocated for every synchronous exception described in the MCAUSE register (Table 3.6 of RISC-V Privileged ISA Spec rev 1.10), with the index of the bit position equal to the value returned in the MCAUSE register. Breakpoint exception has two bit positions corresponding to two possible origins of such exception - EBREAK instruction and Trigger Module.

If corresponding bit is set and the HART is running in M-mode (RUN debug state), then assert of the corresponding exception causes redirection of HART to Debug Mode (to D_HALTED state).

Table 4. Debug M-MODE Exception Redirection Enable Register

Bit(s)	Name	Access	Reset	Description
15:0	redir_en	R/W	0x0	Exception Redirection Enable If the corresponding bit is set, the HART is running in the M-MODE (RUN debug state) and the corresponding exception is asserted, the HART redirects to the D-MODE and halts (transits to the D_HALTED state). During this transition, HART's architectural state changes in accordance with DRUNCONTROL.re_dsbl bit. Bits: • 0 — Instruction address misaligned • 1 — Instruction access fault • 2 — Illegal instruction • 3 — Breakpoint due to EBREAK instruction • 4 — Load address misaligned • 5 — Load access fault • 6 — Store/AMO address misaligned • 7 — Store/AMO access fault • 8 — Environment call • 9 — Breakpoint from Trigger Module • 10..11 — reserved • 12 — Instruction page fault • 13 — Load page fault • 14 — reserved • 15 — Store/AMO page fault
31:16	rsrv	RO	0x0	Reserved

1.6.10. DEREDIREN_S, Debug S-MODE Exception Redirection Enable Register (0x7b9)

DEREDIREN_S has the same layout of bits as DEREDIREN_M.

If the corresponding bit is set and the HART is running in the S-mode (RUN debug state), then assert of the the corresponding exception causes redirection of the HART to the Debug Mode (the D_HALTED state).

1.6.11. DEREDIREN_U, Debug U-MODE Exception Redirection Enable Register (0x7ba)

DEREDIREN_U has the same layout of bits as DEREDIREN_M.

If the corresponding bit is set and the HART is running in the U-mode (RUN debug state), then assert of the the corresponding exception causes redirection of the HART to the Debug Mode (the D_HALTED state).

1.6.12. DIREDIREN, Debug Interrupt Redirection Enable Register (0x7bb)

DIREDIREN has the same layout of bits as MIDELEG and MIP CSRs. That means it has bit positions for all the individual interrupts, and layout of bits matches to those of the MIP register.

If the corresponding bit is set, and the HART is running in corresponding privilege mode (RUN debug state), then assert of the corresponding interrupt causes redirection of the HART to the Debug Mode (the D_HALTED state).

Table 5. Debug Interrupt Redirection Enable Register

Bit(s)	Name	Access	Reset	Description
0	usire	R/W	0	User Software Interrupt Redirection Enable
1	ssire	R/W	0	Supervisor Software Interrupt Redirection Enable
2	rsrv	RO	0	Reserved
3	msire	R/W	0	Machine Software Interrupt Redirection Enable
4	utire	R/W	0	User Timer Interrupt Redirection Enable
5	stire	R/W	0	Supervisor Timer Interrupt Redirection Enable
6	rsrv	RO	0	Reserved
7	mtire	R/W	0	Machine Timer Interrupt Redirection Enable
8	ueire	R/W	0	User External Interrupt Redirection Enable
9	seire	R/W	0	Supervisor External Interrupt Redirection Enable
10	rsrv	RO	0	Reserved
11	meire	R/W	0	Machine External Interrupt Redirection Enable
31:12	rsrv	RO	0x0	Reserved.

1.6.13. DPPC, Debug Program Buffer PC Register (0x7bc)

The behavior of the DPPC register is similar to the one of DPC but it serves transitions between the D_HALTED and the D_RUN states.

Upon entry into the D_HALTED state (both from the RUN and from the D_RUN states), the DPPC is updated with the value of the architectural Program Counter. When resuming to the D_RUN state takes place (during execution of the Program Buffer instructions), the HART's PC is updated with the virtual address, stored in DPPC. That address is used as a starting address of the Program Buffer's procedure when the Program Buffer's address mapping to the HART's address space is implemented and enabled. A debugger may write DPPC to change it via the Abstract Commands which directly access Debug CSRs (without Program Buffer instructions execution).

1.6.14. DSAVE0/1, Debug Save Register 0/1 (0x7be/0x7bf)

These registers are used as intermediate data storage during the Program Buffer's instructions execution.