# RISC-V Control Transfer Records (Smctr/Ssctr)

RISC-V Control Transfer Records TG

Version v0.1.4, 2023-09-15: Draft

# Table of Contents



# Preamble

*This document is in the Development state*

Assume everything can change. This draft specification will change before being accepted as standard, so implementations made to this draft specification will likely not conform to the future standard.

# Copyright and license information

# Contributors

This RISC-V specification has been contributed to directly or indirectly by:

- Beeman Strong <beeman@rivosinc.com>
- Bruce Ableidinger <bruce.ableidinger@sifive.com>
- Snehasish Kumar <snehasishk@google.com>
- Robert Chyla <robert.chyla@sifive.com>
- John Simpson <john.simpson@sifive.com>
- Ved Shanbhogue <ved@rivosinc.com>
- Stefan O'Rear

# Chapter 1. Introduction

A method to record control transfer history is valuable for performance profiling usages, as well as for debug. Control transfers include jump instructions, taken branch instructions, traps, and trap returns. Profiling tools like Linux perf collect control transfer history when sampling, enabling users, and tools like AutoFDO, to identify hot paths for optimization.

Control flow trace capabilities offer very deep transfer history, but the volume of data produced can result in significant performance overheads due to memory bandwidth consumption, buffer management, and decoder overhead. The Control Transfer Records (CTR) extension provides a method to record a limited history in register-accessible internal chip storage, with the intent of dramatically reducing the performance overhead and complexity of collecting transfer history.

CTR defines a circular (FIFO) transfer history array. Recorded transfers are inserted to the head of the array, while older recorded transfers may be overwritten once the array is full. The source PC, target PC, and some optional metadata is stored for each recorded transfer.

The CTR array is accessible through an indirect CSR interface, such that software can specify which logical entry in the array it wishes to read or write. Logical entry 0 is always the youngest recorded transfer, entry 1 is the next youngest, etc.

The machine-level extension **Smctr** encompasses all added CSRs and all behavior modifications for a hart, over all privilege levels. The associated supervisor-level extension **Ssctr** is essentially the same as **Smctr** except for excluding the machine-level CSRs and behaviors that should not be directly visible to supervisor level.

Smctr is dependent on the Smcsrind extension. Ssctr is dependent on the Sscsrind extension.

*Text in warning blocks (such as this one) are comments for the reader, included while the spec is in development. They will be resolved and removed before the spec is frozen.*

# Chapter 2. CSRs

The CTR configuration is selected by the currently active CTR control register, which is mctrcontrol when V=0 (that is, when in M-mode, S/HS-mode, or U-mode), and vsctrcontrol when V=1 (VS-mode or VU-mode). In this document, when referring to the currently active CTR control register, ctrcontrol is used.

*CSR field specifications (such as WARL and WPRI) can be found in the RISC-V Instruction Set Manual vol. II (Privileged Architecture), section 2.3.*

## 2.1. Machine Control Transfer Record Control (mctrcontrol)

The mctrcontrol register is an MXLEN-bit read/write WARL register that enables and configures the CTR capability while V=0.

| **MXLEN-1:48** | **47** | **46** |
|---|---|---|
| 0 | DIROJMPINH | INDOJMPINH |

| **45** | **44** | **43** | **42** | **41** | **40** |
|---|---|---|---|---|---|
| RETINH | CORSWAPINH | DIRJUMPINH | INDJUMPINH | DIRCALLINH | INDCALLINH |

| **39** | **38** | **37** | **36** | **35** | **34** | **33** | **32:20** |
|---|---|---|---|---|---|---|---|
| 0 | ETEN | TKBRINH | NTBREN | TRETINH | INTRINH | EXCINH | 0 |

| **19:16** | **15:13** | **12** | **11** | **10** | **9** | **8** |
|---|---|---|---|---|---|---|
| DEPTH | 0 | LCOFIFRZ | BPFRZ | RASEMU | STE | MTE |

| **7** | **6:3** | **2** | **1** | **0** |
|---|---|---|---|---|
| CLR | 0 | U | S | M |

| **Field** | **Description** |
|---|---|
| M, S, U | Enable transfer recording in the selected privileged mode(s). VS and VU modes can be enabled from vsctrcontrol. |
| CLR | When written to 1, zeroes all implemented CTR entries, regardless of the current CTR depth. Also zeroes mctrstatus. Reads will always return 0 for this bit. |
| MTE | If ETEN=1, enables recording of traps to M-mode when M=0. See External Traps. |
| STE | If ETEN=1, enables recording of traps to S-mode when S=0. See External Traps. |
| RASEMU | Enables RAS Emulation Mode. |

| Field | Description |
| --- | --- |
| BPFRZ | Set mctrstatus.FROZEN on a breakpoint exception. See Freeze. |
| LCOFIFRZ | Set mctrstatus.FROZEN on local counter overflow interrupt (LCOFI). See Freeze. |
| DEPTH[3:0] | Selects the depth of the CTR array. Encodings:<br><br>‘0000 - 16<br><br>‘0001 - 32<br><br>‘0011 - 64<br><br>‘0111 - 128<br><br>‘1111 - 256<br><br>The depth of the CTR array dictates the number of entries to which the hardware will record transfers. For a depth of N, the hardware will record transfers to entries 0..N-1. mireg, mireg2, and mireg3 are read-only 0 when miselect holds a value in N..255.<br><br>Which DEPTH values are supported is implementation-specific, though supported values must be consecutive. An implementation may opt to hardcode some or all of the bits in this field, based on the depth options supported. See Discovery. |
| EXCINH | Inhibit recording of exceptions. See Transfer Type Filtering. |
| INTRINH | Inhibit recording of interrupts. See Transfer Type Filtering. |
| TRETINH | Inhibit recording of trap return instructions. See Transfer Type Filtering. |
| NTBREN | Enable recording of not-taken branch instructions. See Transfer Type Filtering. |
| TKBRINH | Inhibit recording of taken branch instructions. See Transfer Type Filtering. |
| ETEN | Enable recording of external traps, or traps from an enabled mode to a disabled mode. If ETEN=MTE=1, external traps to M-mode will be recorded. If ETEN=STE=1, external traps to S-mode will be recorded. See External Traps. |
| INDCALLINH | Inhibit recording of indirect call instructions. See Transfer Type Filtering. |
| DIRCALLINH | Inhibit recording of direct call instructions. See Transfer Type Filtering. |
| INDJUMPINH | Inhibit recording of indirect jump instructions. See Transfer Type Filtering. |
| DIRJUMPINH | Inhibit recording of direct jump instructions. See Transfer Type Filtering. |
| CORSWAPINH | Inhibit recording of co-routine swap instructions. See Transfer Type Filtering. |
| RETINH | Inhibit recording of function return instructions. See Transfer Type Filtering. |
| INDOJMPINH | Inhibit recording of other indirect jump instructions. See Transfer Type Filtering. |

| Field | Description |
|---|---|
| DIROJMPINH | Inhibit recording of other direct jump instructions. See Transfer Type Filtering. |

M, CLR, and DEPTH are required fields. S is required if S-mode is implemented, and U is required if U-mode is implemented. All other fields are optional.

*When reducing CTR depth, by writing ctrcontrol.DEPTH to a smaller value, software should set ctrcontrol.CLR. This ensures that no transfer state is retained in the now-inaccessible entries above the new depth value.*

All unimplemented fields are read-only 0. Bits 63:60 are reserved for custom extensions.

In RV32, mctrcontrol[63:32] can be accessed via mctrcontrolh.

## 2.2. Supervisor Control Transfer Record Control (sctrcontrol)

The sctrcontrol register provides access to a subset of ctrcontrol. It is accessible from S-mode and VS-mode, such that S-mode accesses are redirected to mctrcontrol and VS-mode accesses are redirected to vsctrcontrol.

Bits 0 and 8 in sctrcontrol are read-only 0. As a result, S-mode and VS-mode do not have access to the M and MTE fields in mctrcontrol. All other ctrcontrol fields are accessible through sctrcontrol.

In RV32, sctrcontrol[63:32] can be accessed via sctrcontrolh.

If S-mode is not implemented, access to sctrcontrol(h) results in an illegal instruction exception.

## 2.3. Virtual Supervisor Control Transfer Record Control (vsctrcontrol)

The vsctrcontrol register is a VSXLEN-bit read/write WARL register that enables and configures the CTR capability while V=1. VS-mode accesses to sctrcontrol are redirected to vsctrcontrol.

| **VSXLEN-1:48** | **47** | **46** |
|---|---|---|
| 0 | DIROJMPINH | INDOJMPINH |

| **45** | **44** | **43** | **42** | **41** | **40** |
|---|---|---|---|---|---|
| RETINH | CORSWAPINH | DIRJUMPINH | INDJUMPINH | DIRCALLINH | INDCALLINH |

| **39** | **38** | **37** | **36** | **35** | **34** | **33** | **32:20** |
|---|---|---|---|---|---|---|---|
| 0 | ETEN | TKBRINH | NTBREN | TRETINH | INTRINH | EXCINH | 0 |

| 19:16 | 15:13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|
| DEPTH | 0 | LCOFIFRZ | BPFRZ | RASEMU | VSTE | 0 |

| 7 | 6:3 | 2 | 1 | 0 |
|---|---|---|---|---|
| CLR | 0 | VU | VS | 0 |

| Field | Description |
|---|---|
| VS, VU | Enable transfer recording in the selected privileged mode(s). |
| VSTE | If ETEN=1, enables recording of traps to VS-mode when VS=0. See External Traps. |
| DEPTH | Provides read-only access to the mctrcontrol.DEPTH field |
| Other field definitions match those of mctrcontrol. The optional fields implemented in vsctrcontrol should match those implemented in mctrcontrol. | |

*Bit positions for VSTE, VS, and VU in vsctrcontrol match those for STE, S, and U in sctrcontrol, respectively. This is to accommodate an (unenlightened) guest OS that is unaware it is running with V=1.*

*vsctrcontrol.DEPTH is a read-only copy of mctrcontrol.DEPTH in order to allow a hypervisor to dictate the depth used by a VM. This simplifies VM migration, by providing the hypervisor a means to require the guest to use a depth supported across all systems in the datacenter.*

*Because vsctrcontrol is active only when V=1, writing a 1 to vsctrcontrol.CLR will affect a clear only when V=1.*

In RV32, vsctrcontrol[63:32] can be accessed via vsctrcontrolh.

If the H extension is not implemented, access to vsctrcontrol(h) results in an illegal instruction exception.

*The TG considered making vsctrcontrol pass-through mctrcontrol fields other than VS, VU, and VSTE. This would simplify behavior on traps and trap returns between V=0 and V=1, since those shared CTR configuration fields would not change. But this would be undesirable for host + guest usages, since it would require switching sctrcontrol on each V transition.*

## 2.4. Machine Control Transfer Record Status (mctrstatus)

The mctrstatus register provides access to CTR status information, and is updated by the hardware when CTR is active (in an enabled privilege mode and not frozen).

| 31:16 | 15 | 14 | 13:8 | 7:0 |
|---|---|---|---|---|
| *WPRI* | FROZEN | WRAP | *WPRI* | WRPTR |

| Field | Description | Access |
|---|---|---|
| WRPTR | Indicates the physical CTR array entry to be written next. Incremented on new transfers recorded, and decremented on qualified returns when ctrcontrol.RASEMU=1. Wraps on increment when the value matches the selected depth-1, and on decrement when the value is 0. Bits above those needed to represent depth-1 (e.g., bits 7:4 for depth=16) are read-only 0. | WARL |
| WRAP | Sticky indication that the WRPTR has wrapped. Set when WRPTR has value depth-1 (where depth is based on ctrcontrol.DEPTH), and a CTR update causes WRPTR to be incremented. Cleared when WRPTR has value zero and a CTR update causes WRPTR to be decremented (which only happens on returns when RASEMU=1), and on CSR writes that set ctrcontrol.CLR. | WARL |
| FROZEN | Inhibit transfer recording. See Freeze. | WARL |

Bits 31:28 are reserved for custom extensions.

*Logical entry 0, accessed via mireg* when miselect=0x200, is always the physical entry preceding the WRPTR entry (WRPTR-1 % depth).*

*Because the mctrstatus register is updated by hardware, writes should be performed with caution. If a multi-instruction read-modify-write to mctrstatus is performed while CTR is active, such that a qualified transfer, or trap that causes CTR freeze, completes between the read and the write, a hardware update could be lost.*

*Exposing the WRPTR provides a more efficient means for synthesizng CTR entries. If a qualified control transfer is emulated, the emulator can simply increment the WRPTR, then write the synthesized record to entry 0. If a qualified function return is emulated while RASEMU=1, the emulator can clear ctrsource.V for entry 0, then decrement the WRPTR.*

*Exposing the WRPTR may also allow support for Linux perf's stack stitching capability.*

## 2.5. Supervisor Control Transfer Record Status (sctrstatus)

The sctrstatus register is an S-mode and VS-mode (writable) alias to mctrstatus.

## 2.6. CSR Listing

| CSR Number | Name | Description |
|---|---|---|
| 0x181 | sctrcontrol | Supervisor Control Transfer Records Control Register |
| 0x182 | sctrcontrolh* | Supervisor Control Transfer Records Control Register upper 32 bits |
| 0x183 | sctrstatus | Supervisor Control Transfer Records Status Register |
| 0x281 | vsctrcontrol | Virtual Supervisor Control Transfer Records Control Register |
| 0x282 | vsctrcontrolh* | Virtual Supervisor Control Transfer Records Control Register upper 32 bits |
| 0x381 | mctrcontrol | Machine Control Transfer Records Control Register |
| 0x382 | mctrcontrolh* | Machine Control Transfer Records Control Register upper 32 bits |
| 0x383 | mctrstatus | Machine Control Transfer Records Status Register |

* For RV32 only

# Chapter 3. Entry Registers

Control transfer records are stored in a CTR array, such that each array entry stores metadata for a single transfer. The CTR array entries are logically accessed via the indirect register access mechanism defined by the Smcsrind/Sscsrind extension. The miselect index range 0x200..0x2FF is reserved for CTR entries 0..255. When miselect holds a value in this range, mireg provides access to ctrsource, mireg2 provides access to ctrtarget, mireg3 provides access to ctrdata, and mireg[456] are read-only zero.

The standard indirect register access rules specified by Smcsrind/Sscsrind apply for CTR. S-mode is able to access CTR entries using the siselect/sireg* interface, with the same behavior described for M-mode above. Similarly, VS-mode is able to access CTR entries using siselect (really vsiselect) and sireg* (really vsireg*). See State Enable Access Control for cases where CTR access from S-mode and VS-mode may be restricted.

For *iselect values in 0x200..0x2FF, vsireg* registers access the same entry register state as mireg* and sireg*, regardless of the privilege mode at the time of access. There is not a separate set of entry registers for V=1.

Undefined bits in CTR entry registers are WPRI. Status fields may be added by future extensions, and software should ignore but preserve any fields that it does not recognize.

*Implementations may opt not to preserve CTR entry state across clock-gated low-power states. A bit to indicate this should be added to the ACPI spec upon ratification.*

## 3.1. Control Transfer Record Source (ctrsource)

The ctrsource register contains the source virtual address (PC) of the recorded transfer. The valid (V) bit is set by the hardware when a transfer is recorded in the selected CTR array entry, and implies that data in ctrsource, ctrtarget, and ctrdata is valid for this entry. All fields are required.

ctrsource is an MXLEN-bit WARL register that must be able to hold all valid virtual addresses. It need not be able to hold an invalid address. When XLEN < MXLEN, software access via *ireg will access only the lower XLEN bits of ctrsource, and implict writes (by recorded transfers) will be zero-extended.

| MXLEN-1:XLEN | XLEN-1:1 | 0 |
|---|---|---|
| 0 | PC[XLEN-1:1] | V |

*CTR entry registers are defined as MXLEN, despite the CSRs used to access them (*ireg*) being XLEN, to ensure that entries recorded in RV64 are not truncated, as a result of CSR Width Modulation, on a transition to RV32.*

*A transfer from an invalid address (which could only occur on an exception) may report a valid address in ctrsource.PC.*

*If we believe a future standard or custom extension may define 1-byte opcodes, then we should not use bit 0 of ctrsource for the V field, nor bit 0 of ctrtarget for MISP. The V bit could be moved to ctrdata, but that would mean software would always need to read ctrdata.*

## 3.2. Control Transfer Record Target (ctrtarget)

The ctrtarget register contains the target (destination) virtual address of the recorded transfer. MISP is optional, it is set by the hardware when the recorded transfer is an instruction whose target or taken/not-taken direction was mispredicted by the branch predictor. MISP is read-only 0 when not implemented.

ctrtarget is an MXLEN-bit WARL register that must be able to hold all valid virtual addresses. It need not be capable of holding an invalid address. When XLEN < MXLEN, software access via *ireg2 will access only the lower XLEN bits of ctrtarget, and implict writes (by recorded transfers) will be zero-extended.

| **MXLEN-1:XLEN** | **XLEN-1:1** | **0** |
|---|---|---|
| 0 | PC[XLEN-1:1] | MISP |

*A transfer to an invalid address may report a valid address in ctrtarget.PC.*

## 3.3. Control Transfer Record Metadata (ctrdata)

The ctrdata register contains metadata for the recorded transfer. This register is required, though all fields within it are optional. Unimplemented fields are read-only 0.

ctrdata is an MXLEN-bit register. When XLEN < MXLEN, software access via *ireg3 will access only the lower XLEN bits of ctrdata.

| **MXLEN-1:32** | **31:16** | **15** | **14:4** | **3:0** |
|---|---|---|---|---|
| *WPRI* | CC | CCV | *WPRI* | TYPE |

| Field | Description | Access |
|---|---|---|
| TYPE[3:0] | Identifies the type of the control flow transfer recorded in the entry. Implementations that do not support this field will report 0.<br><br>0000 - Reserved<br><br>0001 - Exception<br><br>0010 - Interrupt<br><br>0011 - Trap return<br><br>0100 - Not-taken branch<br><br>0101 - Taken branch<br><br>0110 - Reserved<br><br>0111 - Reserved<br><br>1000 - Indirect call<br><br>1001 - Direct call<br><br>1010 - Indirect jump<br><br>1011 - Direct jump<br><br>1100 - Co-routine swap<br><br>1101 - Return<br><br>1110 - Other indirect jump<br><br>1111 - Other direct jump | WARL |
| CCV | Cycle Count Valid. See Cycle Counting. | WARL |
| CC[15:0] | Cycle Count, composed of the Cycle Count Exponent (CCE, in CC[15:12]) and Cycle Count Mantissa (CCM, in CC[11:0]). See Cycle Counting. | WARL |

Bits 14:12 are reserved for custom extensions.

*The TG has debated the merits of including a 3-bit privilege mode field in ctrdata. This would help in cases where multiple privilege modes are recorded, and existing mechanisms for discerning the mode (addressing conventions and kernel mmaps) do not apply or are not available. But it would require some complexity to avoid exposing the presence of virtualization to a VM that is using CTR, and there is question about the value given that existing tools that use similar capabilities from other architectures do not require this information. The TG has thus far opted not to standardize bits for privilege mode, but consensus within the TG has not been reached.*

*Like the Transfer Type Filtering bits in ctrcontrol, the ctrdata.TYPE bits leverage the E-trace itype encodings.*

# Chapter 4. State Enable Access Control

When Smstateen is implemented, the mstateen0.CTR bit controls access to CTR register state from privilege modes less privileged than M-mode. When mstateen0.CTR=0, attempts from privilege modes less privileged than M-mode to access sctrcontrol, vsctrcontrol, sctrstatus, sireg* when siselect is in 0x200..0x2FF, or vsireg* when vsiselect is in 0x200..0x2FF, raise an illegal instruction exception. When mstateen0.CTR=1, accesses to CTR register state behave as described in CSRs and Entry Registers above.

When mstateen0.CTR=0, qualified control transfers executed in privilege modes less privileged than M-mode will continue to implicitly update Entry Registers and mctrstatus.

If the H extension is implemented and mstateen0.CTR=1, the hstateen0.CTR bit controls access to supervisor CTR state (sctrcontrol, sctrstatus, and sireg* when siselect is in 0x200..0x2FF) when V=1. When mstateen0.CTR=1 and hstateen0.CTR=1, VS-mode accesses to supervisor CTR state behave as described in CSRs and Entry Registers above. When mstateen0.CTR=1 and hstateen0.CTR=0, VS-mode accesses to supervisor CTR state that would have completed successfully if hstateen0.CTR was set raise a virtual instruction exception, while others raise an illegal instruction exception.

When hstateen0.CTR=0, qualified control transfers executed while V=1 will continue to implicitly update Entry Registers and mctrstatus.

The CTR bit is bit 55 in mstateen0 and hstateen0.

Bit 60 in mstateen0, defined by Smcsrind, can also restricts access to sireg*/siselect and vsireg*/vsiselect from privilege modes less privileged than M-mode.

# Chapter 5. Behavior

CTR records qualified control transfers. Control transfers are qualified if they meet the following criteria:

- The current privilege mode is enabled
- The transfer type is not inhibited
- mctrstatus.FROZEN is not set

Such qualified transfers update the Entry Registers at logical entry 0, such that older entries are pushed down the stack (the record previously in entry 0 is pushed to entry 1, the record previously in entry 1 is pushed to entry 2, etc). If the CTR array is full, the oldest recorded entry (at depth-1) is overwritten.

Recorded transfers will set the ctrsource.V bit to 1, and will update all implemented record fields.

*In order to collect accurate and representative performance profiles while using CTR, it is recommended that hardware recording of control transfers incurs no added performance overhead, e.g., in the form of retirement or instruction execution restrictions that are not present when CTR is not recording transfers.*

## 5.1. Privilege Mode Transitions

Transfers that change the privilege mode are a special case. What is recorded, if anything, depends on whether the source mode and/or target mode are enabled for recording, and on the transfer type (trap or trap return).

Traps and trap returns between enabled modes are recorded as normal. Traps from a disabled mode to an enabled mode, and trap returns from an enabled mode back to a disabled mode, are partially recorded. In such cases, the PC from the disabled mode (source PC for traps, and target PC for trap returns) is 0. Trap returns from a disabled mode to an enabled mode are not recorded. Traps from an enabled mode to an disabled mode, known as external traps, are not recorded by default, but see External Traps for how they can be recorded.

Debug Mode is always inhibited. Transfers into and out of Debug Mode are never recorded.

## 5.2. Transfer Type Filtering

By default, all control transfers within enabled privileged modes are recorded. Bits 47:32 in ctrcontrol provide a means for software to alter this behavior, by opting out of select transfer types, or opting into non-default types. An implementation may opt to support any combination of transfer type filter bits, or none.

*Because External Traps and Not-taken Branches are not recorded by default, the polarity of the associated enable bits (ETEN and NTBREN) is the opposite of other bits associated with transfer type filtering (TKBRINH, RETINH, etc). Non-default operations require opt-in rather than opt-out. This ensures that default behavior is enabled when transfer type filter bits are set to 0, or are not implemented.*

The transfer type filter bits leverage the type definitions specified in Table 4.4, and described in Section 4.1.1, of the RISC-V Efficient Trace Spec v2.0. An exception is the ETEN bit, discussed in External Traps below.

*For a given implementation, if support for any transfer type filter bit results in reduced software performance, perhaps due to additional retirement restrictions, it is strongly recommended that this reduced performance apply only when the bit is set. Alternatively, support for the bit may be omitted. Maintaining software performance for the default CTR configuration, when all transfer type bits are cleared, is paramount.*

### 5.2.1. External Traps

By default external traps are not recorded, but an optional handshake mechanism exists to allow partial recording. Software running in the target mode of the trap can opt-in to allowing CTR to record traps into that mode even when the mode is inhibited. The MTE, STE, and VSTE bits allow M-mode, S-mode, and VS-mode, respectively, to opt-in. Tools can request to record External Traps by setting the ETEN bit. When an External Trap occurs, only if both ETEN=1 and xTE=1, such that x is the target privilege mode of the trap, will CTR record the trap. In such cases, the target PC is 0.

*The external trap handshake allows both system software and the tools control over what is exposed. M-mode firmware may always set mctrcontrol.MTE=1, but a user-mode profiler may not wish to see any traps. The driver can set sctrcontrol.ETEN=0 to ensure that external traps are not recorded. On the other hand, a VM may wish to record external traps from VU-mode to VS-mode, but the hypervisor may not wish to expose traps from VU/VS-mode to HS-mode. The VM will set ETEN=VSTE=1, but the hypervisor can clear sctrcontrol.STE.*

*No such mechanism exists for recording external trap returns, because the external trap record includes all relevant information, and gives the trap handler (e.g., an emulator) the opportunity to modify the record.*

*Note that external trap recording does not depend on EXCINH/INTRINH, only on ETEN and MTE/STE. Thus, when external traps are enabled, both external interrupts and external exceptions are recorded.*

*STE allows recording of traps from U-mode to S-mode as well as from VS/VU-mode to HS-mode. The hypervisor can flip STE before entering a guest if it wants different behavior for U-to-S vs VS/VU-to-HS. A separate HTE bit could be defined, but ideally it would live in an hctrcontrol CSR, which is otherwise unneeded. We could put it in [ms]ctrcontrol, but the bit position would need special treatment in vsctrcontrol*

*(writable but has no impact on behavior).*

The table below provides details on recording of privilege mode transfers. Standard dependencies on FROZEN and transfer type inhibits also apply, but are not covered by the table.

| **Transfer Type** | **Source Mode** | **Target Mode** | |
|---|---|---|---|
| | | **Enabled** | **Inhibited** |
| **Trap** | **Enabled** | Recorded. | Recorded if ETEN=xTE=1, where x is target mode. Target PC is 0, type is External Trap. |
| | **Inhibited** | Recorded, Source PC is 0. | Not recorded. |
| **Trap Return** | **Enabled** | Recorded. | Recorded, Target PC is 0. |
| | **Inhibited** | Not recorded. | Not recorded. |

If ETEN is implemented, MTE must be implemented as well, as must STE if S-mode is implemented, and VSTE if VS-mode is implemented.

## 5.3. Cycle Counting

The ctrdata register may optionally include a count of CPU cycles elapsed since the prior CTR record. The elapsed cycle count value is represented by the CC field, which has a mantissa component (Cycle Count Mantissa, or CCM) and an exponent component (Cycle Count Exponent, or CCE). The elapsed cycle count can be calculated using the following formula:

```
if (CCE==0):
    return CCM
else:
    return (2^12 + CCM) << CCE-1
endif
```

*When CCE>1, the granularity of the reported cycle count is reduced. For example, when CCE=3, the bottom 2 bits of the cycle counter are not reported, and thus the reported value increments only every 4 cycles. As a result, the reported value represents an undercount of elapsed cycles for most cases (when the unreported bits are non-zero). On average, the undercount will be $(2^{CCE-1}-1)/2$. Software can reduce the average undercount to 0 by adding $(2^{CCE-1}-1)/2$ to each computed cycle count value when CCE>1.*

The CC value is only valid when the Cycle Count Valid (CCV) bit is set. If CCV=0, the CC value may not hold the correct count of elapsed qualified cycles since the last recorded transfer. Qualified cycles are those executed within an enabled privilege mode with FROZEN=0. An implementation must clear CCV for the next recorded transfer upon a write to ctrcontrol, and in any other implementation-specific scenarios where qualified cycles may be not be counted.

An implementation that supports cycle counting must support CCV and all CCM bits, but may

support 0..4 exponent bits in CCE. Unimplemented CCE bits are read-only 0. For implementations that support transfer type filtering, it is recommended to support at least 3 exponent bits. This allows capturing the full latency of most functions, when recording only calls and returns.

The CC value saturates when all implemented bits in CCM and CCE are 1.

## 5.4. RAS Emulation Mode

When ctrcontrol.RASEMU=1, transfer recording behavior is altered to emulate the behavior of a return-address stack (RAS).

- Indirect and direct calls are recorded as normal
- Function returns pop the most recent call, by invalidating entry 0 (setting ctrsource.V=0) and rotating the CTR array, such that (invalidated) entry 0 moves to entry depth-1, and entries 1..depth-1 move to 0..depth-2.
- Co-routine swaps affect both a return and a call. Entry 0 is overwritten.
- Other transfer types are inhibited
- Transfer Type Filtering bits are ignored

*Profiling tools often collect call stacks along with each sample. Stack walking, however, is a complex and often slow process that may require recompilation (e.g., -fno-omit-frame-pointer) to work reliably. With RAS emulation, tools can ask CTR hardware to save call stacks even for unmodified code.*

*CTR RAS emulation has limitations. The CTR array will contain only partial stacks in cases where the call stack depth was greater than the CTR depth, CTR recording was enabled at a lower point in the call stack than main(), or where the CTR array was cleared since main().*

*The CTR stack may be corrupted in cases where calls and returns are not symmetric, such as with stack unwinding (e.g., setjmp/longjmp, C++ exceptions), where stale call entries may be left on the CTR stack, or user stack switching, where calls from multiple stacks may be intermixed.*

*As described in Cycle Counting, when CCV=1, the CC field provides the elapsed cycles since the prior CTR entry was recorded. This introduces implementation challenges when RASEMU=1 because, for each recorded call, there may have been several recorded calls (and returns which "popped" them) since the prior remaining call entry was recorded. The implication is that returns that pop a call entry not only do not reset the cycle counter, but instead add the CC field from the popped entry to the counter. For simplicity, an implementation may opt to record CCV=0 for all calls, or those whose parent call was popped, when RASEMU=1.*

## 5.5. Freeze

When mctrstatus.FROZEN=1, transfer recording is inhibited. This bit can be set by hardware, as

described below, or by software.

When ctrcontrol.LCOFIFRZ=1 and a local counter overflow interrupt (LCOFI) traps, mctrstatus.FROZEN is set by the CPU. This inhibits CTR recording until software clears FROZEN. The LCOFI trap itself is not recorded.

*Freeze on LCOFI ensures that the execution path leading to the sampled instruction (xepc) is preserved, and that the local counter overflow interrupt (LCOFI) and associated Interrupt Service Routine (ISR) do not displace any recorded transfer history state. It is the responsibility of the ISR to clear FROZEN before xRET, if continued control transfer recording is desired.*

*LCOFI refers only to architectural traps directly caused by a local counter overflow. If a local counter overflow interrupt is recognized without a trap, for instance by reading mip, FROZEN is not automatically set.*

Similarly, on a breakpoint exception with ctrcontrol.BPFRZ=1, FROZEN is set by the CPU. The breakpoint exception itself is not recorded.

*Breakpoint exception refers to synchronous exceptions with a cause value of Breakpoint (3), regardless of source (ebreak, c.ebreak, Sdtrig); it does not include entry into Debug Mode, even in cores where this is implemented as an exception.*

# Chapter 6. Discovery

Software can discover supported CTR array depth values using the following method:

- Write ‘0000 to ctrcontrol.DEPTH, then read back the value. The value read represents the minimum supported depth.
- Write ‘1111 to ctrcontrol.DEPTH, then read back the value. The value read represents the maximum supported depth.

All depths between the minimum and maximum are supported.

Software can discover implemented optional ctrcontrol fields by writing all 1s to all defined fields, then reading the value back. Unimplemented fields are read-only 0.

Software can discover implemented optional CTR entry fields by writing all 1s to all defined fields in the Entry Registers at entry 0, then reading them back. Unimplemented fields are read-only 0.