RISC-V Platform-Level Interrupt Controller Specification

Version 1.0, 09/2021

Table of Contents

Change Log

Date	Version	Description
2021/9/17	v1.0	The initial version of this specification

Copyright and license information

This RISC-V PLIC specification is

Chapter 1. Introduction

This document contains the RISC-V platform-level interrupt controller (PLIC) specification (was removed from RISC-V Privileged Spec v1.11-draft) , which defines an interrupt controller specifically designed to work in the context of RISC-V systems. The PLIC multiplexes various device interrupts onto the external interrupt lines of Hart contexts, with hardware support for interrupt priorities.
This specification defines the general PLIC architecture and the operation parameters. PLIC supports up-to 1023 interrupts (0 is reserved) and 15872 contexts, but the actual number of interrupts and context depends on the PLIC implementation. However, the implement must adhere to the offset of each register within the PLIC operation parameters. The PLIC which claimed as PLIC-Compliant standard PLIC should follow the implementations mentioned in sections below.

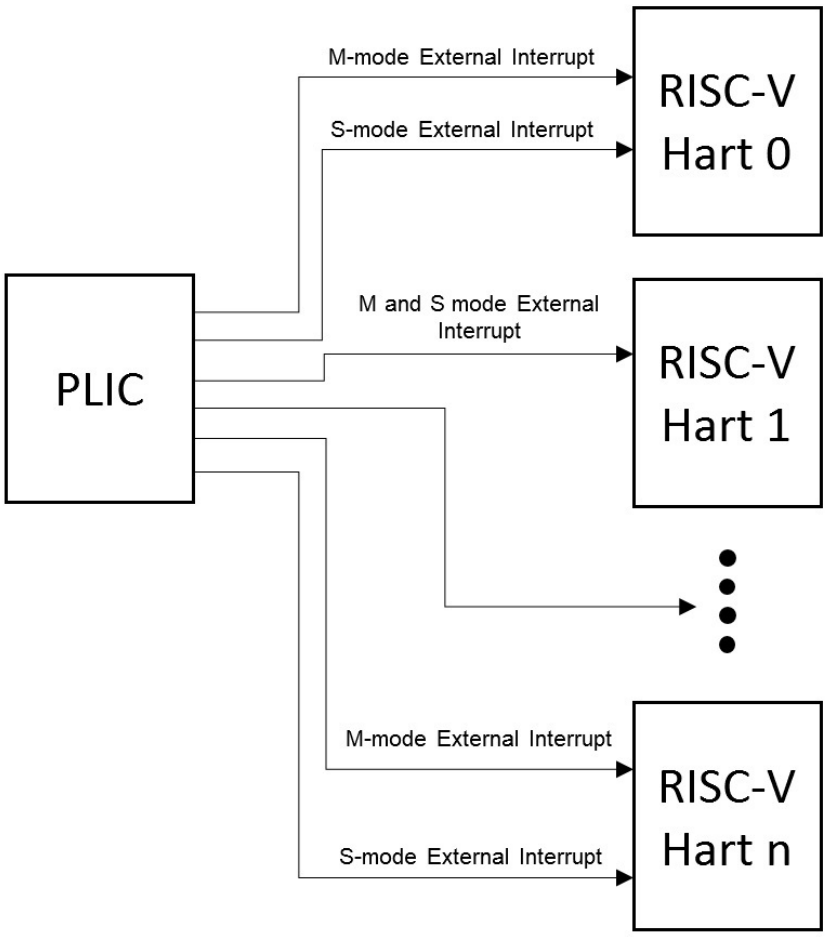

Figure 1. RISC-V PLIC Interrupt Architecture Block Diagram

Chapter 2. RISC-V PLIC Operation Parameters

General PLIC operation parameter register blocks are defined in this spec, those are:

- **Interrupt Priorities registers:**
  The interrupt priority for each interrupt source.
- **Interrupt Pending Bits registers:**
  The interrupt pending status of each interrupt source.
- **Interrupt Enables registers:**
  The enablement of interrupt source of each context.
- **Priority Thresholds registers:**
  The interrupt priority threshold of each context.
- **Interrupt Claim registers:**
  The register to acquire interrupt source ID of each context.
- **Interrupt Completion registers:**
  The register to send interrupt completion message to the associated gateway.

  Below is the figure of PLIC Operation Parameter Block Diagram,

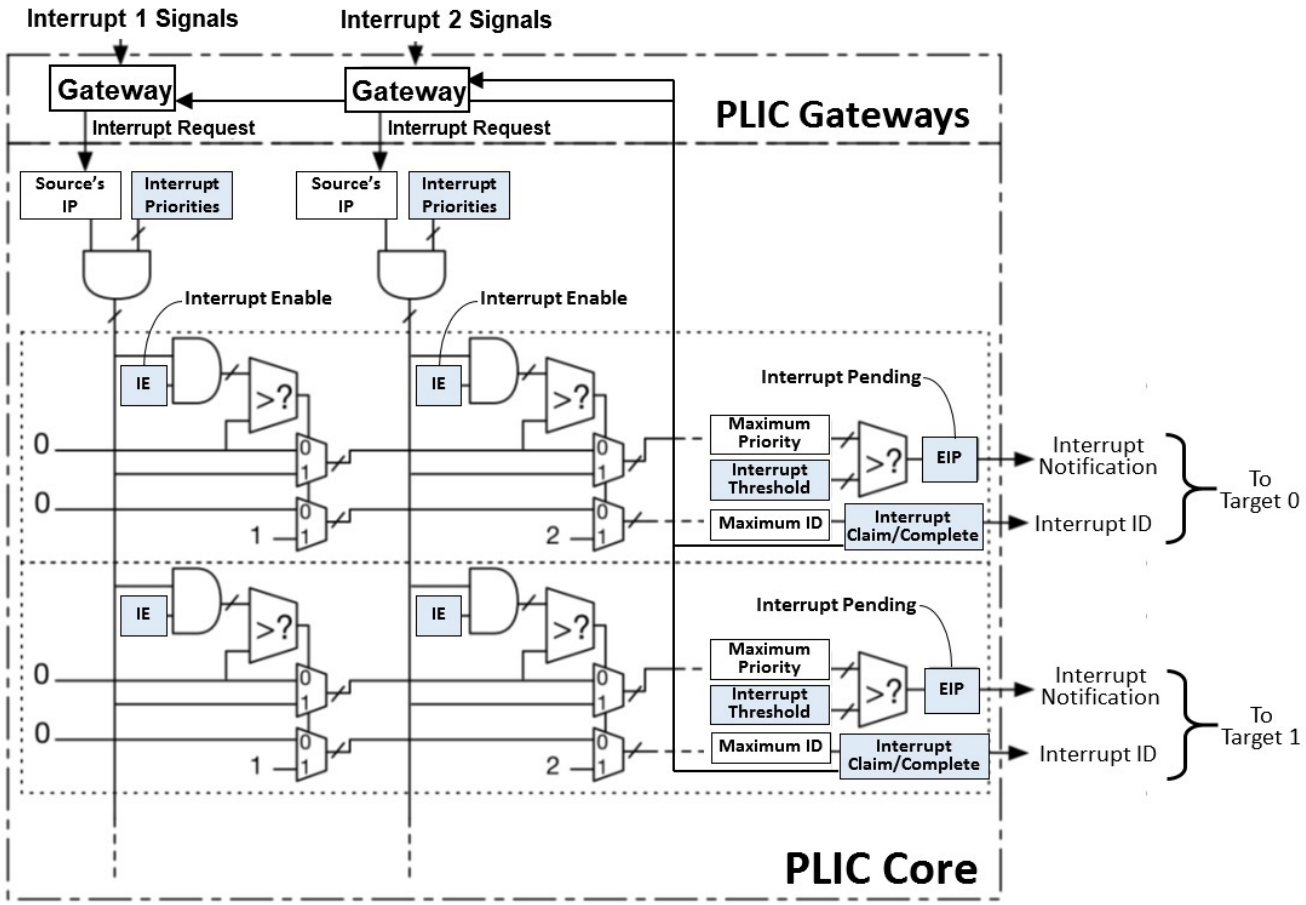

Figure 2. PLIC Operation Parameter Block Diagram

Chapter 3. Memory Map

The base address of PLIC Memory Map is platform implementation-specific.

PLIC Memory Map

```
base + 0x000000: Reserved (interrupt source 0 does not exist)
base + 0x000004: Interrupt source 1 priority
base + 0x000008: Interrupt source 2 priority
...
base + 0x000FFC: Interrupt source 1023 priority
base + 0x001000: Interrupt Pending bit 0-31
base + 0x00107C: Interrupt Pending bit 992-1023
...
base + 0x002000: Enable bits for sources 0-31 on context 0
base + 0x002004: Enable bits for sources 32-63 on context 0
...
base + 0x00207C: Enable bits for sources 992-1023 on context 0
base + 0x002080: Enable bits for sources 0-31 on context 1
base + 0x002084: Enable bits for sources 32-63 on context 1
...
base + 0x0020FC: Enable bits for sources 992-1023 on context 1
base + 0x002100: Enable bits for sources 0-31 on context 2
base + 0x002104: Enable bits for sources 32-63 on context 2
...
base + 0x00217C: Enable bits for sources 992-1023 on context 2
...
base + 0x1F1F80: Enable bits for sources 0-31 on context 15871
base + 0x1F1F84: Enable bits for sources 32-63 on context 15871
base + 0x1F1FFC: Enable bits for sources 992-1023 on context 15871
...
base + 0x1FFFFC: Reserved
base + 0x200000: Priority threshold for context 0
base + 0x200004: Claim/complete for context 0
base + 0x200008: Reserved
...
base + 0x200FFC: Reserved
base + 0x201000: Priority threshold for context 1
base + 0x201004: Claim/complete for context 1
...
base + 0x3FFF000: Priority threshold for context 15871
base + 0x3FFF004: Claim/complete for context 15871
base + 0x3FFF008: Reserved
...
base + 0x3FFFFFC: Reserved
```

Sections below describe the control register blocks of PLIC operation parameters.

Chapter 4. Register Width

The memory map register width is in 32-bit.

Chapter 5. Interrupt Priorities

If PLIC supports Interrupt Priorities, then each PLIC interrupt source can be assigned a priority by writing to its 32-bit memory-mapped `priority` register. A priority value of 0 is reserved to mean ''never interrupt'' and effectively disables the interrupt. Priority 1 is the lowest active priority while the maximum level of priority depends on PLIC implementation. Ties between global interrupts of the same priority are broken by the Interrupt ID; interrupts with the lowest ID have the highest effective priority.

The base address of Interrupt Source Priority block within PLIC Memory Map region is fixed at 0x000000.

PLIC Register Block Name	Function	Register Block Size in Byte	Description
Interrupt Source Priority	Interrupt Source Priority #0 to #1023	1024 * 4 = 4096(0x1000) bytes	This is a continuously memory block which contains PLIC Interrupt Source Priority. Total 1024 Interrupt Source Priority in this memory block. Interrupt Source Priority #0 is reserved which indicates it does not exist.

PLIC Interrupt Source Priority Memory Map

```
0x000000: Reserved (interrupt source 0 does not exist)
0x000004: Interrupt source 1 priority
0x000008: Interrupt source 2 priority
...
0x000FFC: Interrupt source 1023 priority
```

Chapter 6. Interrupt Pending Bits

The current status of the interrupt source pending bits in the PLIC core can be read from the pending array, organized as 32-bit register. The pending bit for interrupt ID N is stored in bit (N mod 32) of word (N/32). Bit 0 of word 0, which represents the non-existent interrupt source 0, is hardwired to zero.

A pending bit in the PLIC core can be cleared by setting the associated enable bit then performing a claim.

The base address of Interrupt Pending Bits block within PLIC Memory Map region is fixed at 0x001000.

PLIC Register Block Name	Function	Register Block Size in Byte	Description
Interrupt Pending Bits	Interrupt Pending Bit of Interrupt Source #0 to #N	1024 / 8 = 128(0x80) bytes	This is a continuously memory block contains PLIC Interrupt Pending Bits. Each Interrupt Pending Bit occupies 1-bit from this register block.

PLIC Interrupt Pending Bits Memory Map

```
0x001000: Interrupt Source #0 to #31 Pending Bits
...
0x00107C: Interrupt Source #992 to #1023 Pending Bits
```

Chapter 7. Interrupt Enables

Each global interrupt can be enabled by setting the corresponding bit in the `enables` register. The `enables` registers are accessed as a contiguous array of 32-bit registers, packed the same way as the `pending` bits. Bit 0 of enable register 0 represents the non-existent interrupt ID 0 and is hardwired to 0. PLIC has 15872 Interrupt Enable blocks for the contexts.

The `context` is referred to the specific privilege mode in the specific Hart of specific RISC-V processor instance. For example, in an 4-core system with 2-way SMT, you have 8 harts and probably at least two privilege modes per hart: machine mode and supervisor mode. (ref: github.com/torvalds/linux/blob/5bfc75d92efd494db37f5c4c173d3639d4772966/Documentation/devicetree/bindings/interrupt-controller/sifive%2Cplic-1.0.0.yaml)

How PLIC organizes interrupts for the contexts (Hart and privilege mode) is out of RISC-V PLIC specification scope, however it must be spec-out in vendor's PLIC specification.

The base address of Interrupt Enable Bits block within PLIC Memory Map region is fixed at 0x002000.

PLIC Register Block Name	Function	Register Block Size in Byte	Description
Interrupt Enable Bits	Interrupt Enable Bit of Interrupt Source #0 to #1023 for 15872 contexts	(1024 / 8) * 15872 = 2031616(0x1f0000) bytes	This is a continuously memory block contains PLIC Interrupt Enable Bits of 15872 contexts. Each Interrupt Enable Bit occupies 1-bit from this register block and total 15872 Interrupt Enable Bit blocks

PLIC Interrupt Enable Bits Memory Map

```
0x002000: Interrupt Source #0 to #31 Enable Bits on context 0
...
0x00207F: Interrupt Source #992 to #1023 Enable Bits on context 0
0x002080: Interrupt Source #0 to #31 Enable Bits on context 1
...
0x0020FF: Interrupt Source #992 to #1023 Enable Bits on context 1
0x002100: Interrupt Source #0 to #31 Enable Bits on context 2
...
0x00217F: Interrupt Source #992 to #1023 Enable Bits on context 2
0x002180: Interrupt Source #0 to #31 Enable Bits on context 3
...
0x0021FF: Interrupt Source #992 to #1023 Enable Bits on context 3
...
...
...
0x1F1F80: Interrupt Source #0 to #31 on context 15871
...
0x1F1F80: Interrupt Source #992 to #1023 on context 15871
```

Chapter 8. Priority Thresholds

PLIC provides context based `threshold register` for the settings of a interrupt priority threshold of each context. The `threshold register` is a WARL field. The PLIC will mask all PLIC interrupts of a priority less than or equal to `threshold`. For example, a`threshold` value of zero permits all interrupts with non-zero priority.

The base address of Priority Thresholds register block is located at 4K alignement starts from offset 0x200000.

PLIC Register Block Name	Function	Register Block Size in Byte	Description
Priority Threshold	Priority Threshold for 15872 contexts	4096 * 15872 = 65011712(0x3e00000) bytes	This is the register of Priority Thresholds setting for each conetxt

PLIC Interrupt Priority Thresholds Memory Map

```
0x200000: Priority threshold for context 0
0x201000: Priority threshold for context 1
0x202000: Priority threshold for context 2
0x203000: Priority threshold for context 3
...
...
...
0x3FFF000: Priority threshold for context 15871
```

Chapter 9. Interrupt Claim Process

The PLIC can perform an interrupt claim by reading the `claim/complete` register, which returns the ID of the highest priority pending interrupt or zero if there is no pending interrupt. A successful claim will also atomically clear the corresponding pending bit on the interrupt source.
The PLIC can perform a claim at any time and the claim operation is not affected by the setting of the priority threshold register.
The Interrupt Claim Process register is context based and is located at (4K alignement + 4) starts from offset 0x200000.

PLIC Register Block Name	Function	Register Block Size in Byte	Description
Interrupt Claim Register	Interrupt Claim Process for 15872 contexts	4096 * 15872 = 65011712(0x3e00000) bytes	This is the register used to acquire interrupt ID for each conetxt

PLIC Interrupt Claim Process Memory Map

```
0x200004: Interrupt Claim Process for context 0
0x201004: Interrupt Claim Process for context 1
0x202004: Interrupt Claim Process for context 2
0x203004: Interrupt Claim Process for context 3
...
...
...
0x3FFF004: Interrupt Claim Process for context 15871
```

Chapter 10. Interrupt Completion

The PLIC signals it has completed executing an interrupt handler by writing the interrupt ID it received from the claim to the `claim/complete` register. The PLIC does not check whether the completion ID is the same as the last claim ID for that target. If the completion ID does not match an interrupt source that is currently enabled for the target, the completion is silently ignored.
The Interrupt Completion registers are context based and located at the same address with Interrupt Claim Process register, which is at (4K alignement + 4) starts from offset 0x200000.

PLIC Register Block Name	Registers	Register Block Size in Byte	Description
Interrupt Completion Register	Interrupt Completion for 15872 contexts	4096 * 15872 = 65011712(0x3e00000) bytes	This is register to write to complete Interrupt process

PLIC Interrupt Completion Memory Map

```
0x200004: Interrupt Completion for context 0
0x201004: Interrupt Completion for context 1
0x202004: Interrupt Completion for context 2
0x203004: Interrupt Completion for context 3
...
...
...
0x3FFF004: Interrupt Completion for context 15871
```

Contributors

Abner Chang <abner.chang@hpe.com>
Campbell He
Chunchi Che
Drew Barbier <drew@sifive.com>
Jeff Scheel <jeff@riscv.org>
Jessica Clarke
Palmer Dabbelt <palmer@dabbelt.com>
Rongcui Dong
Yan <phantom@zju.edu.cn>


RISC-V