NSI-TEXE

NSITEXE NS31A User Manual

NSITEXE, Inc.

NRV31210004-00-E Revision 3.0.0

Index

About This Document

This document is a User Manual for NS31A.

It provides the hardware configuration and logic specifications for NS31A, as well as information on interfaces with external modules.
The target readers of the document are designers and software programmers who will implement systems with NS31A mounted.

For details of the specifications, see the NSITEXE NS31A Technical Reference Manual.

Notices and Disclaimers

Target Design

This document is intended for the following hardware design kit.

- NRV31210003

Relevant Documents

- NSITEXE NS31A Technical Reference Manual (NRV31220000)
- NSITEXE NS31A Integration Manual (NRV31220002)

- NSITEXE NS31A Hardware Safety Manual (NRV31810001)

Reference Documents

Item #	Document	Revision Used	Comment
1	The RISC-V Instruction Set Manual Volume I: Unprivileged ISA	20190608-Base-Ratified	
2	The RISC-V Instruction Set Manual Volume II: Privileged Architecture	20211105-signoff	
3	The RISC-V Instruction Set Manual Volume II: Privileged Architecture	draft-20221004-28b46de	Only for Smrnmi specification
4	RISC-V Debug Support	1.0.0-STABLE	
5	RISC-V Bitmanip Extension	1.0.0	
6	RISC-V Advanced Core Local Interruptor Specification	1.0-rc4	
7	RISC-V Platform-Level Interrupt Controller Specification	1.0.0	
8	AMBA® AXI™ and ACE™ Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite	ARM IHI 0022D (ID102711)	
9	AMBA® APB Protocol Version: 2.0 Specification	ARM IHI 0024C (ID041610)	
10	AMBA® AHB Protocol Specification	ARM IHI 0033B.b (ID102715)	

Abbreviations

Abbreviation	Description
BTB	Branch Target Buffer
CMU	Core Management Unit
CSR	Control and Status Register
DBG	Debug Subsystem
DFT	Design For Test
DLM	Data Local Memory
ECC	Error Correction Code
EDC	Error Detection and Correction
hart	Hardware Thread
ACLINT	Advanced Core Local Interruptor
PLIC	Platform-Level Interrupt Controller
ILM	Instruction Local Memory
ISA	Instruction Set Architecture
PMP	Physical Memory Protection
TAP	Test Access Port

Chapter 1. NSITEXE NS31A Core Overview

This chapter describes the overview of NSITEXE NS31A Processor.
NS31A is a 32-bit processor core optimized for control-specific microcontrollers.
The core supports the instruction sets of RISC-V, which is an open architecture. Therefore, this product can be used in various eco-systems.
The advanced, high-performance AXI configuration or the small-area AHB configuration can be selected using a configuration parameter.

1.1. Features

NS31A has the following features.

NS31A Core	• 32-bit processor core with a 4-stage pipeline structure and some RISC-V instruction sets
	• Base Integer Instruction Set (RV32I or RV32E) (When configuration parameter P_RV32E is 1, RV32E is enabled.)
	• Compressed Instructions (RV32C) (Only when configuration parameter P_RV32C is 1)
	• Bit Manipulation Instructions (RV32B) (Only when configuration parameter P_RV32B is 1)
	• Single-Precision Floating Point (RV32F) (Only when configuration parameter P_FPU is 1)
	• Branch prediction mechanism with a Branch Target Buffer (BTB) (Only when configuration parameter P_BTB is 1)
	• 2-stage privilege modes (M-mode and U-mode)
	• 2 configuration options for the integer multiplier (Can be changed using configuration parameter P_MUL_SMALL.)
	• 2 configuration options for the integer divider (Can be changed using configuration parameter P_DIV_SMALL.)
Instruction Cache (IC)	• 32-Byte Line 2-way Set Associative Cache
	• Variable capacity (Using configuration parameter P_CACHE_SIZE_IC, the capacity can be selected from none to 32 KB.)

Instruction Local Memory (ILM)	• Allocated only in the AXI configuration. • Can be allocated in a given address space. • Variable capacity (Using configuration parameter P_ADDR_SIZE_ILM, the capacity can be selected from none to 8 MB.)
Data Local Memory (DLM)	• Allocated only in the AXI configuration. • Can be allocated in a given address space. • Variable capacity (Using configuration parameter P_ADDR_SIZE_DLM, the capacity can be selected from 4 KB to 8 MB.)
Interrupt Control Units (ACLINT/PLIC)	• External Interrupts with a priority processing mechanism that can support up to 255 factors • Non-Maskable Interrupts that can be requested from the outside • Software Interrupts which can directly be requested by accessing a register • Timer interrupt from Internal Timer
Core Management Unit (CMU)	• System control and observation functions using register interfaces
Bus Interface	• In the case of the AXI configuration ◦ System Bus Master interface (32-bit AXI4 Master) assuming access to resources in SoC ◦ External Peripheral Bus Master interface (32-bit APB4 Master) assuming access to registers of peripheral modules, etc. ◦ Front Port Bus Slave interface (32-bit AXI4 Slave) assuming access from an external master to internal RAM resources and registers • In the case of the AHB configuration ◦ Instruction Memory Bus (32-bit AHB5 lite Master) for fetching instructions ◦ Data Memory Bus (32-bit AHB5 lite Master) for access data in memory and registers in SoC ◦ Peripheral Bus Slave (32-bit APB4 Slave) for access to registers in NS31A

Functional Safety	• Physical Memory Protection for protecting address spaces (The number of entries can be changed using configuration parameter P_PMPNUM.) • RAM ECC for protecting RAM data (When configuration parameter P_REDUNDANT is 1 or 2) • Bus Error Detection and Correction (EDC) for protecting buses (Only when configuration parameter P_REDUNDANT is 2 in the AXI configuration) • Dual Core Lock-Step for realizing redundancy (only when configuration parameter P_REDUNDANT is 2)
Debug Unit	• Controls the execution of the hart • Debug resource access by a JTAG debugger or via a bus • Compliant with RISC-V External Debug Specification

1.2. Function Block Diagram

Figure 1 shows the block diagram of the AXI configuration of the NSITEXE NS31A processor.

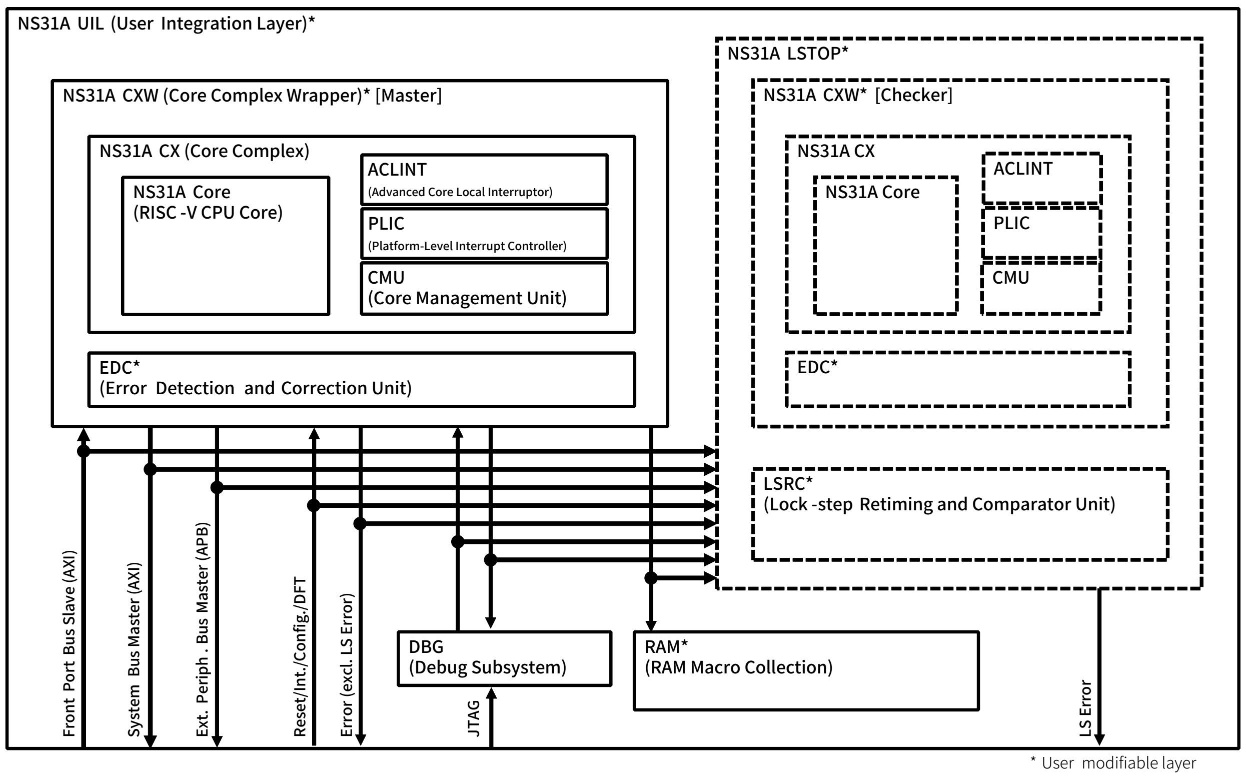
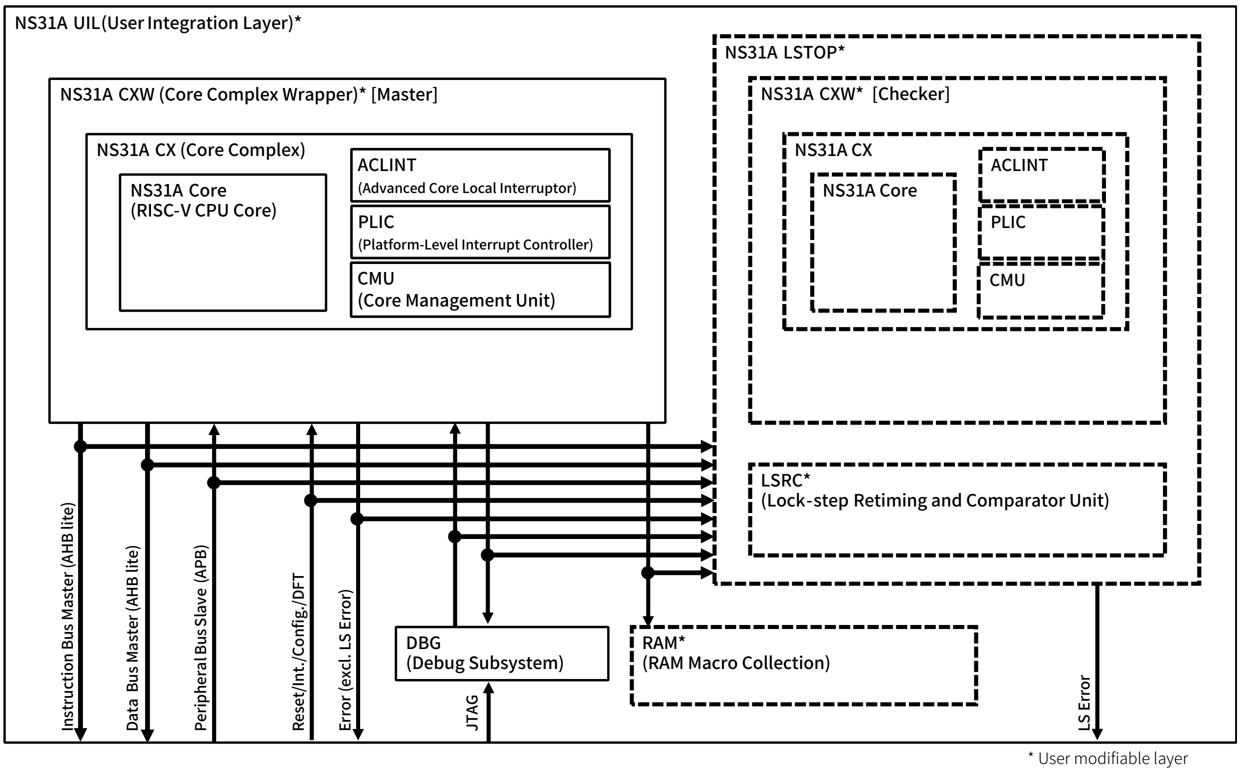


Figure 1. NS31A Block Diagram (AXI configuration)

Figure 2 shows the block diagram of the AHB configuration of the NSITEXE NS31A processor.



NS31A UIL(User Integration Layer)*
NS31A CXW (Core Complex Wrapper)* [Master]
NS31A CX (Core Complex)
NS31A Core (RISC-V CPU Core)
ACLINT (Advanced Core Local Interruptor)
PLIC (Platform-Level Interrupt Controller)
CMU (Core Management Unit)
NS31A LSTOP*
NS31A CXW* [Checker]
NS31A CX
NS31A Core
ACLINT
PLIC
CMU
LSRC* (Lock-step Retiming and Comparator Unit)
DBG (Debug Subsystem)
RAM* (RAM Macro Collection)
Instruction Bus Master (AHB lite)
Data Bus Master (AHB lite)
Peripheral Bus Slave (APB)
Reset/Int./Config./DFT
Error (excl. LS Error)
JTAG
LS Error
* User modifiable layer

Figure 2. NS31A Block Diagram (AHB configuration)

The following layers are implementation samples, which can be modified by the user.

NS31A User Integration Layer (UIL)	Top layer of NS31A for use by the user.
NS31A Lock-Step Top (LSTOP)	Layer to be added when NS31A is used in the Dual Core Lock-Step. Implemented only when configuration parameter P_REDUNDANT is 2.
NS31A Core Complex Wrapper (CXW)	Wrapper layer for integrating NS31A CX and NS31A EDC.
Error Detection and Correction Unit (EDC)	Unit for performing bus, RAM ECC, and parity processing. It consists of RAM ECC and BUS EDC. RAM ECC is implemented when configuration parameter P_REDUNDANT is 1 or 2. BUS EDC is implemented only when configuration parameter P_REDUNDANT is 2 in the AXI configuration. BUS EDC is not implemented in the AHB configuration.
Lock-Step Retiming and Comparator Unit (LSRC)	Unit for allocating a retiming (delay) FF and a comparator for a Lock-Step configuration. Implemented only when configuration parameter P_REDUNDANT is 2.

RAM	RAM Macro module. This layer and the layers below it need to be implemented by the user according to the process to be used.

The following layers and the layers below them cannot be modified by the user.

NS31A Core Complex (CX)	Layer on which the NS31A Core and peripheral units are integrated.
NS31A Core	RISC-V CPU Core of NS31A.
Advanced Core Local Interruptor (ACLINT)	Unit for controlling the core local interrupts.
Platform-Level Interrupt Controller (PLIC)	Unit for controlling the external interrupts.
Core Management Unit (CMU)	Unit for controlling and monitoring the operations in the core.
Debug Subsystem (DBG)	Unit in which debug mechanisms are integrated.

Chapter 2. Memory Map

This chapter describes the memory map of NS31A and the behavior of each memory access operation.

2.1. Address Map

2.1.1. Address Regions

The memory and register resources in NS31A are allocated in the address regions shown in Table 1, respectively.

In the AXI configuration, regions other than these are External System Bus regions. Access from NS31A to any of these regions causes access from the System Bus Master. The resources in the address regions shown in Table 1 can be accessed from an external bus master via the Front Port Bus Slave.

In the AHB configuration, Instruction Local Memory and Data Local Memory are not mounted in NS31A. Also, the External Peripheral Bus is not implemented. NS31A makes instruction access to all addresses from the Instruction Bus Master and data access from the Data Bus Master. Only register regions can be accessed from an external bus master only via the Peripheral Bus Slave.

Access may result in an error response.

In the AXI configuration, error responses are classified into SLVERR and DECERR. The response classifications are indicated in the following descriptions.
In the AHB configuration, error responses are not classified.

SLVERR and DECERR errors in the AXI configuration and bus errors in the AHB configuration are not classified by software. An error caused by a Load instruction is notified as a non-blocking load bus error NMI, and an error caused by a Store or AMO instruction is notified as a non-blocking store bus error NMI. For details, see Section 3.12.3.

In NS31A, IP is designed with incorporation into a SoC system in mind. Therefore, there is no specific absolute address on the address map of each resource.
The start address of each region and the size of some address regions will be given by specifying them in relevant parameters when IP is implemented. The start address specified in the parameter needs to be aligned with the address region size.
For details, see Section 12.1.

Table 1. Address Region List

Region	**Start Address (Parameter)**	**Size (Parameter)**	**Description**
Instruction Local Memory	P_ADDR_BASE_ILM (*1)	P_ADDR_SIZE_ILM (*1)	The size can be changed.
Data Local Memory	P_ADDR_BASE_DLM (*1)	P_ADDR_SIZE_DLM (*1)	The size can be changed.
External Peripheral Bus	P_ADDR_BASE_PER (*1)	P_ADDR_SIZE_PER (*1)	The size can be changed.
Debug Subsystem Register	P_ADDR_BASE_DBG	Fixed to 16 KB.	
Hart Context Register	P_ADDR_BASE_HCR	Fixed to 32 KB.	
ACLINT Register	P_ADDR_BASE_ACLINT	Fixed to 64 KB.	

Region	Start Address (Parameter)	Size (Parameter)	Description
CMU Register	P_ADDR_BASE_CMU	Fixed to 4 KB.	
Internal Register	P_ADDR_BASE_REG	Fixed to 4 KB.	
PLIC Register	P_ADDR_BASE_PLIC	Fixed to 4 MB.	

*1 In the AHB configuration, these configuration parameters are disabled.

2.1.1.1. Instruction Local Memory Region Address Map

Table 2 shows the address map of the Instruction Local Memory (ILM) Region.
The external bus master of NS31A can also access this region via the Front Port Bus Slave interface.

Table 2. ILM Region Address Map (P_ADDR_BASE_ILM + Offset)

Start Address	End Address	Region	Size
+0x00_0000	+size(P_ADDR_SIZE_ILM)-1	ILM	size(P_ADDR_SIZE_ILM)

This region is not implemented if the AHB configuration is selected (configuration parameter P_ADDR_SIZE_ILM is disabled).
Even when the AXI configuration is selected, the region is not implemented if NS31A is configured with configuration parameter P_ADDR_SIZE_ILM set to 0.

2.1.1.2. Data Local Memory Region Address Map

Table 3 shows the address map of the Data Local Memory (DLM) Region.
The external bus master of NS31A can also access this region via the Front Port Bus Slave interface.

Table 3. DLM Region Address Map (P_ADDR_BASE_DLM + Offset)

Start Address	End Address	Region	Size
+0x00_0000	+size(P_ADDR_SIZE_DLM)-1	DLM	size(P_ADDR_SIZE_DLM)

This region is not implemented if the AHB configuration is selected (configuration parameter P_ADDR_SIZE_DLM is disabled).

2.1.1.3. External Peripheral Bus Region Address Map

Table 4 shows the address map of the External Peripheral Bus Region.
Access to this region is access to the External Peripheral Bus. For details of the External Peripheral Bus Master, see Section 4.1.2.
The external bus master of NS31A can also access this region via the Front Port Bus Slave interface.
Do not allocate an instruction code in this region.

Table 4. External Peripheral Bus Region Address Map (P_ADDR_BASE_PER + Offset)

Start Address	End Address	Region	Size
+0x00_0000	+size(P_ADDR_SIZE_PER)-1	External Peripheral Bus	size(P_ADDR_SIZE_PER)

This region is not implemented if the AHB configuration is selected (configuration parameter P_ADDR_SIZE_PER is disabled). Even when the AXI configuration is selected, the region is not implemented if NS31A is configured with configuration parameter P_ADDR_SIZE_PER set to 0.

2.1.1.4. Internal Register Region Address Map

Table 5 shows the address map of the Internal Register Region.
This is a region where internal control registers are allocated. The external bus master of NS31A can also access this region via the Front Port Bus Slave in the case of the AXI configuration or via the Peripheral Bus Slave interface in the case of AHB configuration.

Table 5. Internal Register Region Address Map (P_ADDR_BASE_REG + Offset)

Start Address	End Address	Region	Size
+0x0000	+0x0FFF	Memory Mapped Control Register	4KB

When a reserved region in the Region in Table 5 is accessed, a bus error response is returned (DECERR in the case of the AXI configuration).
If P_BUS_PROT0_PRIV parameter is set to 1, the following error responses are returned.

- If the Region is accessed from NS31A CORE with the U-mode privilege, a bus error response is returned.
- In the AXI configuration, if the Region is accessed from the Front Port Bus Slave with the AxPROT[0] = 0 (Unprivileged) privilege, a bus error response (SLVERR) is returned.
- In the AHB configuration, if the Region is accessed from the Peripheral Bus Slave with the PPROT[0] = 0 (Normal) privilege, an error response is returned.

2.1.1.5. CMU Register Region Address Map

Table 6 shows the address map of the CMU Register Region.
This is a region where internal control registers are allocated.
The external bus master of NS31A can also access this region via the Front Port Bus Slave in the case of the AXI configuration or via the Peripheral Bus Slave interface in the case of AHB configuration.

Table 6. CMU Register Region Address Map (P_ADDR_BASE_CMU + Offset)

Start Address	End Address	Region	Size
+0x0000	+0x0FFF	CMU control register	4KB

When a reserved region in the Region in Table 6 is accessed, 0 is read for a read access and a write access is ignored.
If P_BUS_PROT0_PRIV parameter is set to 1, the following error responses are returned.

- If the Region is accessed from NS31A CORE with the U-mode privilege, a bus error response is returned.
- In the AXI configuration, if the Region is accessed from the Front Port Bus Slave with the AxPROT[0] = 0 (Unprivileged) privilege, a bus error response (SLVERR) is returned.
- In the AHB configuration, if the Region is accessed from the Peripheral Bus Slave with the PPROT[0] = 0 (Normal) privilege, an error response is returned.

2.1.1.6. ACLINT Register Region Address Map

Table 7 shows the address map of the ACLINT Register Region.
This is a region where internal control registers are allocated. The external bus master of NS31A can also access this region via the Front Port Bus Slave in the case of the AXI configuration or via the Peripheral Bus Slave interface in the case of AHB configuration.

Table 7. ACLINT Register Region Address Map (P_ADDR_BASE_ACLINT + Offset)

Start Address	End Address	Region	Size
+0x0000	+0xFFFF	ACLINT control register	64KB

When an unused region in the Region in Table 7 is accessed, 0 is read for a read access and a write access is ignored. If P_BUS_PROT0_PRIV parameter is set to 1, the following error responses are returned.

- If the Region is accessed from NS31A CORE with the U-mode privilege, a bus error response is returned.
- In the AXI configuration, if the Region is accessed from the Front Port Bus Slave with the AxPROT[0] = 0 (Unprivileged) privilege, a bus error response (SLVERR) is returned.
- In the AHB configuration, if the Region is accessed from the Peripheral Bus Slave with the PPROT[0] = 0 (Normal) privilege, an error response is returned.

2.1.1.7. PLIC Register Region Address Map

Table 8 shows the address map of the PLIC Register Region.
This is a region where internal control registers are allocated. The external bus master of NS31A can also access this region via the Front Port Bus Slave in the case of the AXI configuration or via the Peripheral Bus Slave interface in the case of AHB configuration.

Table 8. PLIC Register Region Address Map (P_ADDR_BASE_PLIC + Offset)

Start Address	End Address	Region	Size
+0x000000	+0x3FFFFF	PLIC control register	4MB

When an unused region in the Region in Table 8 is accessed, 0 is read for a read access and a write access is ignored. If P_BUS_PROT0_PRIV parameter is set to 1, the following error responses are returned.

- If the Region is accessed from NS31A CORE with the U-mode privilege, a bus error response is returned.
- In the AXI configuration, if the Region is accessed from the Front Port Bus Slave with the AxPROT[0] = 0 (Unprivileged) privilege, a bus error response (SLVERR) is returned.
- In the AHB configuration, if the Region is accessed from the Peripheral Bus Slave with the PPROT[0] = 0 (Normal) privilege, an error response is returned.

2.1.1.8. Hart Context Register Region Address Map

Table 9 shows the address map of the Hart Context Register Region.
This is a region where internal control registers are allocated. The external bus master of NS31A can also access this region via the Front Port Bus Slave in the case of the AXI configuration or via the Peripheral Bus Slave interface in the

case of AHB configuration.

Table 9. Hart Context Register Region Address Map (P_ADDR_BASE_HCR + Offset)

Start Address	End Address	Region	Size
+0x0000	+0x7FFF	Hart context register	32KB

The following access to this region causes a bus error response (SLVERR in the case of the AXI configuration).

- Access to an unused area in this region

In the case of the AXI configuration, the following access to this region causes a bus error response (SLVERR).

- Access from NS31A CORE with the U-mode privilege (only in case of P_BUS_PROT0_PRIV=1)
- Instruction fetch access from NS31A CORE
- Access from the Front Port Bus Slave with AxPROT[0] set to 0 (Unprivileged) (only in case of P_BUS_PROT0_PRIV=1)
- Access from the Front Port Bus Slave with AxPROT[2] set to 1 (Instruction)

In the case of the AHB configuration, the following access to this region causes a bus error response.

- Access from the Peripheral Bus Slave with the PPROT[0] = 0 (Normal) privilege (only in case of P_BUS_PROT0_PRIV=1)
- Access from the Peripheral Bus Slave with the PPROT[2] = 1 (Instruction) privilege

There are also other conditions that cause a bus error response.
For details, see Section 3.8.2.

2.1.1.9. Debug Subsystem Register Region Address Map

Table 10 shows the address map of the Debug Subsystem Register Region.
This region can be accessed from the external bus master of NS31A via the Front Port Bus Slave in the case of the AXI configuration or via the Peripheral Bus Slave interface in the case of AHB configuration.

Table 10. Debug Subsystem Register Region Address Map (P_ADDR_BASE_DBG + Offset)

Start Address	End Address	Region	Size
+0x0000	+0x00FF	Safe Zero Address	256B
+0x0100	+0x03FF	**Reserved**	**768B**
+0x0400	+0x05FF	Debug module register	1KB
+0x0600	+0x2FFF	**Reserved**	**10KB**
+0x3000	+0x3FFF	**Error Device**	**4KB**

Access to Safe Zero Address region

If the Safe Zero Address region in this region is accessed from NS31A CORE, Front Port Bus Slave, or Peripheral Bus Slave, A write access is ignored and a read data is always zero. No bus error response occurs.

Access to Reserved region

If the Reserved region in this region is accessed from NS31A CORE, Front Port Bus Slave, or Peripheral Bus Slave, a bus error response is returned (SLVERR in the case of the AXI configuration).

Access to Error Device region

If the Error Device region in this region is accessed from NS31A CORE, Front Port Bus Slave, or Peripheral Bus Slave, a bus error response is returned (SLVERR in the case of the AXI configuration).

Access to Debug module register

If an unused region in the Debug module register is read, an undefined value is returned. A write access is ignored. In such cases, bus access is completed normally and functions other than bus access are not affected.
Accessing the Debug module register is restricted by the Debug Authentication function.
If the input value of the dbg_auth signal is 0, any access to this region causes a bus error response (DECERR in the case of the AXI configuration).
For details of the Debug Authentication function, see NSITEXE NS31A Technical Reference Manual.

Access with the U-mode privilege

When P_BUS_PROT0_PRIV parameter is set to 1, the following bus errors are returned.

- In the case of the AXI configuration, if this region is accessed from NS31A Core with the U-mode privilege, a bus error response (DECERR) is returned.
- If access is made from the Front Port Bus Slave with the AxPROT[0] = 0 (Unprivileged) privilege, a bus error response (SLVERR) is returned.
- In the case of the AHB configuration, if access is made from the Peripheral Bus Slave with the PPROT[0] = 0 (Normal) privilege, a bus error response is returned.

2.1.2. Bus access routing

In NS31A, the region that can be accessed differs depending on the master.
Table 11 shows the region that can be accessed from each master in the case of the AXI configuration. If an inaccessible region is accessed, a bus error response (DECERR) is returned.

Table 11. Space That Can Be Accessed from Each Master(AXI configuration)

Region	CPU	Front Port Bus Slave
Instruction Local Memory	✓	✓
Data Local Memory	✓	✓
External Peripheral Bus	✓	✓
Debug Subsystem Register	✓	✓
Hart Context Register	✓	✓
ACLINT Register	✓	✓
CMU Register	✓	✓
Internal Register	✓	✓
PLIC Register	✓	✓
External System Bus	✓	-

In the AHB configuration, all address regions can be accessed as a flat address space.
Instruction access is issued from the Instruction Bus Master Interface, and data access is issued from the Data Bus Master Interface.
Instruction Local Memory, Data Local Memory, and External Peripheral Bus are not implemented.
The regions of ACLINT Register, CMU Register, PLIC Register, Internal Register, Hart Context Register, and Debug Subsystem Register can be accessed as part of address spaces from the Peripheral Bus Slave via the external bus.
Note, however, that instructions cannot be allocated in the address regions of ACLINT Register, CMU Register, PLIC Register, Internal Register, and Hart Context Register.
If an inaccessible region is accessed, a bus error response is returned.

Bus access from DBG is prohibited. During debugging, use Abstract Command to access from the CPU. For details, see NSITEXE NS31A Technical Reference Manual.

2.1.2.1. Address Region Attribute

Each address region has the following attributes defined independently.

Cacheable

This indicates whether the instruction cache is enabled or disabled in the address region. However, whether data is actually cached or not differs depending on individual factors (e.g. in the Debug Mode, when the cache is disabled, etc.).

Side effect

This indicates whether the read/write operation from/to an address region has a side effect. If there is no side effect, idempotence is guaranteed for the region (generally, I/O registers and control registers have a side effect).
It is impossible to combine a write operation to a region where idempotence is not guaranteed.

Table 12 shows whether each address region has these attributes in the case of the AXI configuration.
In the case of the AHB configuration, all address regions are "cacheable" and "have a side effect".

Table 12. Whether Each Address Region Has These Attributes (AXI Configuration)

Region	Cacheable	Side effect
Instruction Local Memory	Yes	No
Data Local Memory	Yes	No
External Peripheral Bus	Yes	Yes
Debug Subsystem Register	Yes	Yes
Hart Context Register	*1	Yes
ACLINT Register	*1	Yes
CMU Register	*1	Yes
Internal Register	*1	Yes
PLIC Register	*1	Yes
External System Bus	Yes	Yes

*1 This is a cacheable region, but no instruction codes can be allocated to it.

For the support status of mutual exclusion instructions (RV32A) in each address region, see Section 3.10.

Access to an address region with a side effect is controlled so that an access request will not be split nor combined.

2.2. RAM

This chapter describes the RAM mounted in NS31A.

Table 13 shows the list of RAMs in NS31A.
ILM and DLM are not mounted in the AHB configuration. IC is not mounted when configuration parameter P_CACHE_SIZE_IC is 0, and ILM is not mounted when configuration parameter P_ADDR_SIZE_ILM is 0.
The functions related to unmounted RAMs that are described below are disabled.

Table 13. RAM List

RAM	NS31A
Instruction Cache (IC)	0B~32KB
Instruction Local Memory (ILM)	0B~8MB
Data Local Memory (DLM)	4KB~8MB

2.2.1. Features

Access

The bus masters in NS31A and the masters connected to the Front Port can access all RAMs (ILM and DLM) that have addresses.
The address allocation is determined by the parameters shown in Section 2.1.1.

Test interface

All the RAMs including caches have a test interface that can be accessed from the software.

2.2.1.1. Test interface

Table 14 shows the test interface of each RAM in NS31A.

direct memory access

Implies the test that is conducted via a normal memory access route.

in-direct memory access via control register

Implies the access that is performed via a control register.

Table 14. RAM Test interface

RAM	Address mapping	Test interface for Data RAM
ILM	mapped	direct memory access
DLM	mapped	direct memory access
Inst. Cache Tag	unmapped	in-direct memory access via control register
Inst. Cache Data	unmapped	in-direct memory access via control register

For details of the control registers that access the internal RAM in the Instruction Cache, see Section 3.8.3.
To access the internal RAM of a cache, see Section 2.3.3.4.

2.3. Cache Control

This chapter describes the cache system of NS31A.
NS31A has the following cache.

- Level-1: Instruction Cache
  (Only when configuration parameter P_CACHE_SIZE_IC ≥ 1)

The Level-1 Instruction Cache caches the instruction fetch target address and neighboring area. Note that it does not cache literals.
In NS31A, the Level-1 Instruction Cache is enabled for all the address regions when P_CACHE_SIZE_IC ≥ 1. However, when the hart state is the Debug mode, it is always disabled, regardless of the address or cache setting.

2.3.1. Features

Each cache has the following control functions.
These functions are controlled via control registers or Cache Clear signals, which are external pin inputs.

- Enabling/disabling caches
- Cache flush
  - Register setting
  - External pin input
- Diagnosis access

These functions are not available when configured with P_CACHE_SIZE_IC set to 0.

2.3.2. Registers

2.3.2.1. Instruction Cache registers

The cache control registers of the Instruction Cache and the control registers that controls access to the internal RAM of the cache are provided as the memory mapped registers of the processor. See Section 3.8.3.

2.3.3. Operations

2.3.3.1. Cache enable/disable

To control whether to enable or disable a cache, perform the following procedure.

- Enabling a cache
  1. Write 1 to bit 0 (Cache Enable) of the ml1cachesysctrl register.
  2. Execute the FENCE instruction.
  3. When a write complete response is returned, the enabling is completed.

- Disabling a cache
  1. Write 0 to bit 0 (Cache Enable) of the ml1cachesysctrl register.
  2. Execute the FENCE instruction.
  3. When a write complete response is returned, the disabling is completed.

2.3.3.2. Cache clear (register setting)

To control the clearing of a cache by using a register, perform one of the following procedures.

- Clear control via register access
  1. Write 1 to bit 0 (Flush Operation) of the ml1cacheoperation register.

> - If the ml1cacheoperation register is accessed using the store instruction when P_CACHE_SIZE_IC is set to 0, a bus error occurs.
>   The value of P_CACHE_SIZE_IC can be checked using the CFGCSIZEIC register of the CMU.
>   Make this check as necessary before accessing ml1cacheoperation.
> - When writing data to the ml1cacheoperation register using the store instruction, execute the FENCE instruction to wait for the completion of clearing the cache.

- Clear control using the CMU register access
  1. Read the CFGCSIZEIC register, and check the value of configuration parameter P_CACHE_SIZE_IC.
  2. When configuration parameter P_CACHE_SIZE_IC $\geq$ 1, exercise clear control by taking steps 3 to 5 below.
  3. Write 1 to bit 16 (COMPCCLR) of the STATCCLR register to clear the completion flag to 0.
  4. Write 1 to bit 0 (CCLRREQ) of the CTRLCCLR register.
  5. When bit 16 (COMPCCLR) of the STATCCLR register becomes 1, the cache has been completely cleared.

2.3.3.3. Cache clear (external input pin)

The caches of NS31A can be cleared by asserting the external pin bccx_cache_clr_req.
In this cache clearing, the completion of clearing the cache is notified to the outside by asserting ncc_cache_clr_ack.
Do not negate the asserted ncc_cache_clr_req until ncc_cache_clr_ack is asserted.
When executing the cache clearing again, have 5 or more cycles of the clk clock after the negation of ncc_cache_clr_ack before asserting ncc_cache_clr_req.

Under ideal conditions, the cache clearing is completed within 20 cycles after the request signal is asserted.
The number of cycles until the cache is actually cleared is determined according to the internal state of NS31A.
It is possible to check whether the cache has been completely cleared by checking the assertion of ncc_cache_clr_ack or using the STATCCLR.COMPCCLR flag.

This function is disabled when no cache is mounted in the configuration.

2.3.3.4. Diagnosis access

The caches of NS31A can be accessed by the registers for testing. To access the internal RAM of a cache, perform the following procedure.
(Only when configuration parameter P_CACHE_SIZE_IC ≥ 1)

Procedure for write-access to the internal RAM of a cache

1. Set the access target RAM and address in the L1CRATGT register.
2. Set the lower 32 bits of Write Data in the L1CRADAT0 register.
3. To access DataRAM, set the upper 32 bits of Write Data in the L1CRADAT1 register.
4. Write 2 or 3 to the RAC[1:0] bits of the L1CRACTR register.

Procedure for read-access to the internal RAM of a cache

1. Set the access target RAM and address in the L1CRATGT register.
2. Write 1 to the RAC[1:0] bits of the L1CRACTRL register.
3. Read the lower 32 bits of Read Data from the L1CRADAT0 register.
4. To access DataRAM, read the upper 32 bits of Read Data from the L1CRADAT1 register.

Chapter 3. NS31A Core

This chapter describes the NS31A Core that performs a series of arithmetic processing in NS31A.

3.1. Overview

The NS31A Core is a processor core designed in accordance with the RISC-V Instruction Set Architecture (ISA) standard.
In the AXI configuration, it contains unique memories (ILM and DLM) that share address spaces and NS31A fetches instruction codes from these memories or from outside the core.
ILM and DLM are not mounted in the AHB configuration. Instruction codes are fetched from outside the core.

3.1.1. Execution Control Part

The NS31A Core adopts a 4-stage pipeline structure with a branch prediction mechanism*1 that uses a Branch Target Buffer (BTB).
The Execution Control Part performs instruction execution, exception handling, interrupt processing, and error handling, among others.

*1 The branch prediction mechanism is enabled only when configuration parameter P_BTB is 1.

The instruction sets of NS31A are based on the RISC-V architecture.
For the instruction sets, see Section 3.9.

3.1.2. Hardware Thread

The hardware thread (hart) in NS31A is equivalent to the hart defined in the RISC-V and equipped with thread contexts (PC, general-purpose register, and CSR).
For details of each CSR, see Section 3.7.
NS31A is equipped with one hart.

3.2. Physical Memory Protection (PMP)

This section describes memory protection using physical memory protection (PMP).

3.2.1. Features

PMP is a mechanism that monitors an access address from the bus master to each memory region and detects illegal access to an address out of the specified range.
NS31A has the PMP implemented to protect the NS31A Core from instruction or data access.
The PMP is implemented only when configuration parameter P_PMPNUM ≥ 1.

The PMP function is not available when configured with P_PMPNUM set to 0. Access is always permitted.

PMP has the following functions.

Address check

- Address width: 32 bits
- Minimum region granularity
  - Data: 4 bytes (32 bits)
  - Instruction: 32 bytes (256 bits)
    (For details of the minimum region granularity for instruction fetch, see Section 3.12.6.)
- Number of entries to be checked: Max. 16
  (The number of entries can be changed using configuration parameter P_PMPNUM.)
- Support the following address range settings specified in RISC-V Priv. Arch. V1.11 specifications.
  - TOR: Top of range
    Entry 0: 0x0 $\leq$ (target addr) < PMPADDR0
    Entry n: PMPADDRm $\leq$ (target addr) < PMPADDRn, (n=1-15, m=n-1)
  - NA4: Naturally aligned four-byte region
  - NAPOT: Naturally aligned power-of-two region
    2^n byte region ($3 \leq n \leq 32$), which is set in PMPADDR

Entry lock

- Each PMP entry has a Lock bit (pmpncfg.L). Locking an entry protects it from being changed.
- Check the R/W/X permissions even in M-mode when locking an entry.

Entry match, priority

- If the access address is within the address range of the entry, determine whether the access is permitted or not for the entry.
- If there are multiple entries that match an address (Overlapping), an entry with the smallest entry number has priority.

- In U-mode, always determine whether access is permitted for the entry. If there are no matching entries, access is illegal.
- In M-mode, determine whether the following accesses are permitted.
    - Instruction fetch
        - Determine whether access is permitted using only the PMP entries for which pmpncfg.L is set to 1.
        - Access is permitted if there are no valid entries whose address matches.
    - Data access
        - When mstatus.MPRV = 1 and mstatus.MPP = 0 (access with U-mode privilege)
            - Determine whether access is permitted using all PMP entries.
            - Access is not permitted if there are no valid entries whose address matches.
        - Other than the above
            - Determine whether access is permitted using only the PMP entries for which pmpncfg.L is set to 1.
            - Access is permitted if there are no valid entries whose address matches.

Behavior performed if an illegal access occurs

- An exception occurs
- Error notification

  In NS31A, an exception is generated if an illegal access occurs. For details of the an exception that is generated, see Section 3.4. An error notification is issued by asserting the err_pmp signal for one cycle.

3.2.1.1. Entry match and priority

An example of entry match and priority is shown in Figure 3.
Figure 3 shows a case where there is overlapping in entry settings. An entry with a smaller entry number has higher priority.
This example assumes that access is permitted in the register setting if no PMP entries match.

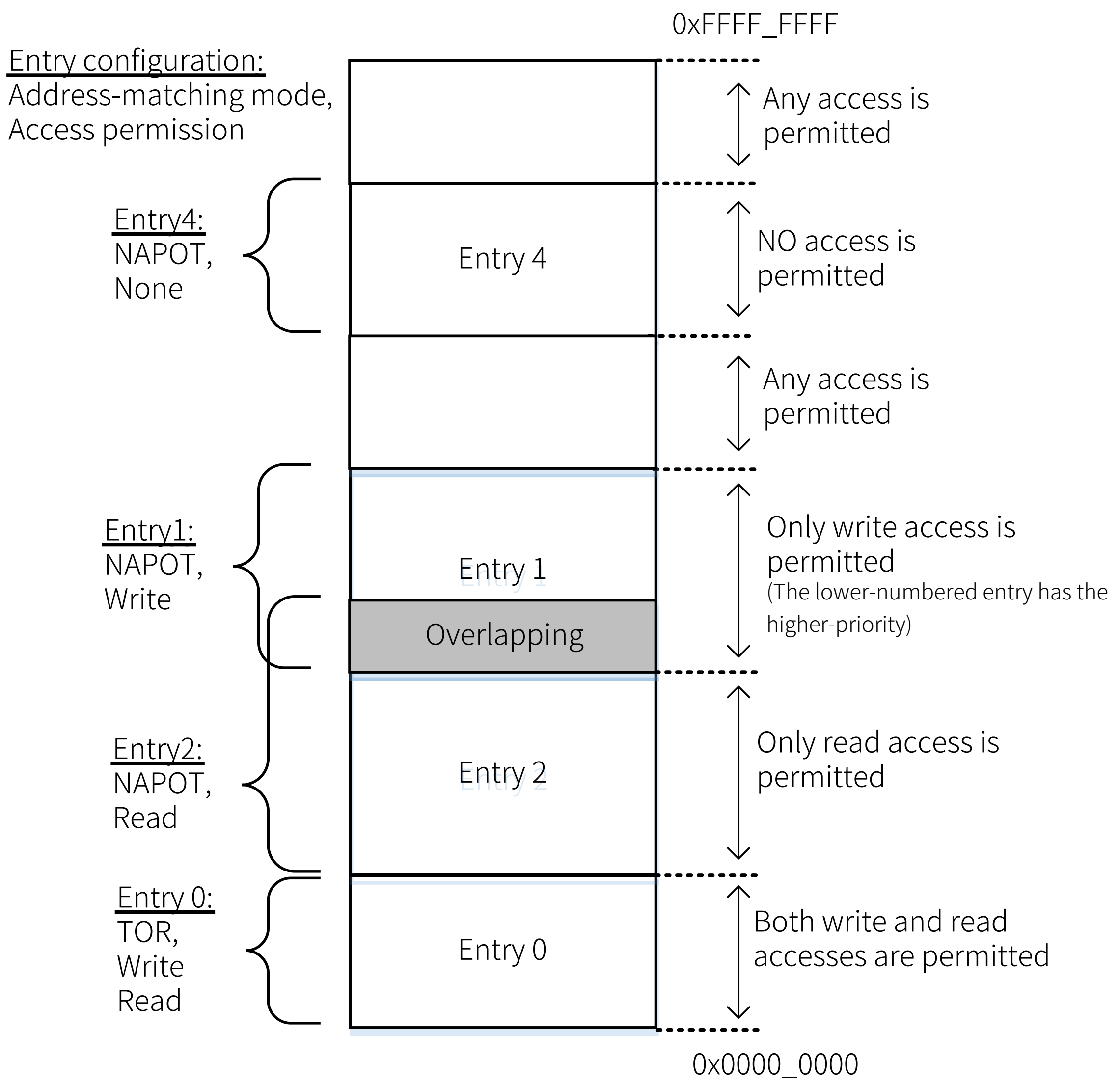


Figure 3. Example of overlapping PMP entries

3.2.1.2. Behavior in the Debug mode

While the hart is in the Debug mode, PMP does not perform access protection, regardless of the register settings.

3.2.2. Registers

For the PMP registers, see Section 3.7.2.7.

3.3. Operating States

NS31A has the following three types of states as a state indicating the operating state of each layer.

- System-level states
- Core-level states
- Hart-level states

Table 15 shows the correspondence between layers and states.

Table 15. Operating States Level

State	Layer
System-level states	NS31A
Core-level states	NS31A Core
Hart-level states	hart0

The following shows the meaning of each state.

System-level states

System-level states are the states that NS31A has, which indicate the operating states of the core (and hart) and peripheral modules.

Core-level states

Core-level states are the states that the core (NS31A Core) has, which indicate the operating states of the core.

Hart-level states

Hart-level states are the states that the hardware thread (hart) has, which indicate the operating states of the hart.

3.3.1. System-level states

System-level states are the states that NS31A (NS31A CX) has, which indicate the operating states of NS31A Core Complex.
System-level states are controlled via the System Control pins on the NS31A CX layer. Be sure to assert the System Control pins only in a state that is indicated as a transition condition in Figure 4. For details of the control pins, see Table 161.
The operating states of NS31A CX are output to the NS31A CX output pin ncc_stat.
The following explains each state and shows a state transition diagram.
For details of the System-level State transition timing, see Section 5.3.

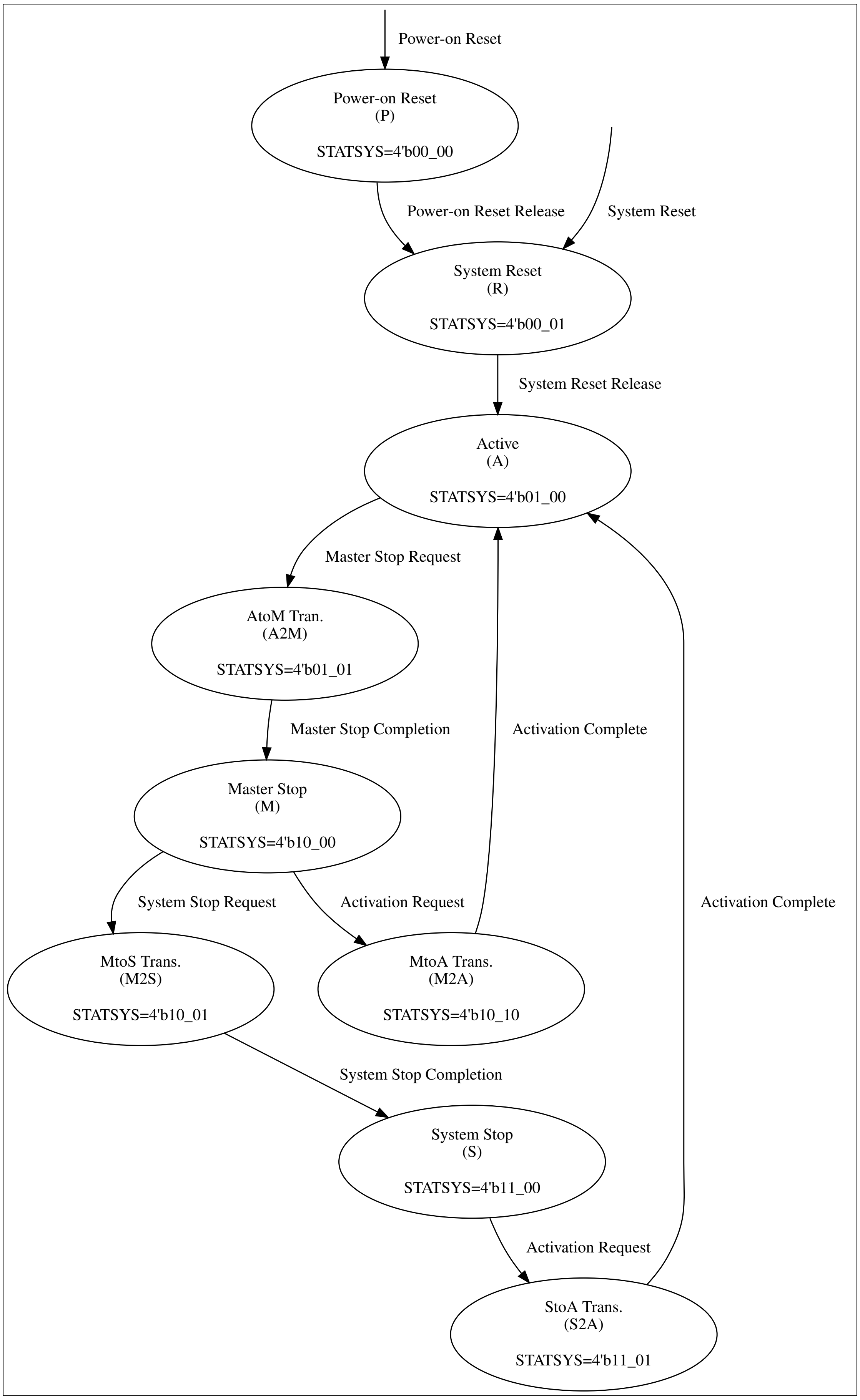

Power-on Reset
Power-on Reset (P) STATSYS=4'b00_00
Power-on Reset Release
System Reset
System Reset (R) STATSYS=4'b00_01
System Reset Release
Active (A) STATSYS=4'b01_00
Master Stop Request
AtoM Tran. (A2M) STATSYS=4'b01_01
Master Stop Completion
Activation Complete
Master Stop (M) STATSYS=4'b10_00
System Stop Request
Activation Request
Activation Complete
MtoS Trans. (M2S) STATSYS=4'b10_01
MtoA Trans. (M2A) STATSYS=4'b10_10
System Stop Completion
System Stop (S) STATSYS=4'b11_00
Activation Request
StoA Trans. (S2A) STATSYS=4'b11_01

Figure 4. System-level states Transition diagram

Power-on Reset

The state of power-ON reset. No functions of NS31A can be used.

System Reset

The state of system reset. Except some system management functions, error functions, and debug-related functions, functions cannot be used.

Active

Active state. All functions of NS31A can be used.

Master Stop

Master stop state. The execution of programs is suspended, and the issuance of all bus transactions is stopped. It is possible to access registers and RAM in NS31A from the outside of NS31A via the Front Port Bus Slave (AXI configuration) or Peripheral Bus Slave (AHB configuration).

System Stop

System stop state. In addition to the stop of the Master Stop state, it is impossible to access the Front Port Bus Slave (AXI configuration) or Peripheral Bus Slave (AHB configuration) and an access request to a register or RAM cannot be accepted according to the bus protocol.
In this state, it is guaranteed that there are no unprocessed bus transactions in NS31A. Therefore, it is possible to execute the system reset of NS31A alone without affecting the external bus.

AtoM Transition

A state where the transition from Active to Master Stop is in progress.

MtoS Transition

A state where the transition from Master Stop to System Stop is in progress.

MtoA Transition

A state where the transition from Master Stop to Active is in progress.

StoA Transition

A state where the transition from System Stop to Active is in progress.

The following shows conditions for each state transition.

Power-on Reset

Perform the Power-on Reset according to the procedure described in Section 5.2.1.

System Reset

Perform the System Reset according to the procedure described in Section 5.2.2.

Power-on Reset Release

Release the Power-on Reset according to the procedure described in Section 5.2.1.

System Reset Release

Release the System Reset according to the procedure described in Section 5.2.2.

Master Stop Request

Assert ncc_mststop_req to request all bus masters to stop.

Master Stop Completion

The transition to this state is made when all bus masters have stopped.
At this time, assert ncc_mststop_ack to notify the outside about the completion.

System Stop Request

Assert ncc_sysstop_req to request the system to stop.

System Stop Completion

The transition to this state is made when the system has stopped.
At this time, assert ncc_sysstop_ack to notify the outside about the completion.

Activation Request

Assert ncc_active_req to request the return from the bus master stop or system stop.

Activation Complete

The transition to this state is made when a return from the bus master stop or system stop is completed.
At this time, assert ncc_active_ack to notify the outside about the completion.

3.3.2. Core-level states

Core-level states are the states that the core (NS31A Core Complex) has.
Core-level states are controlled together with System-level states.

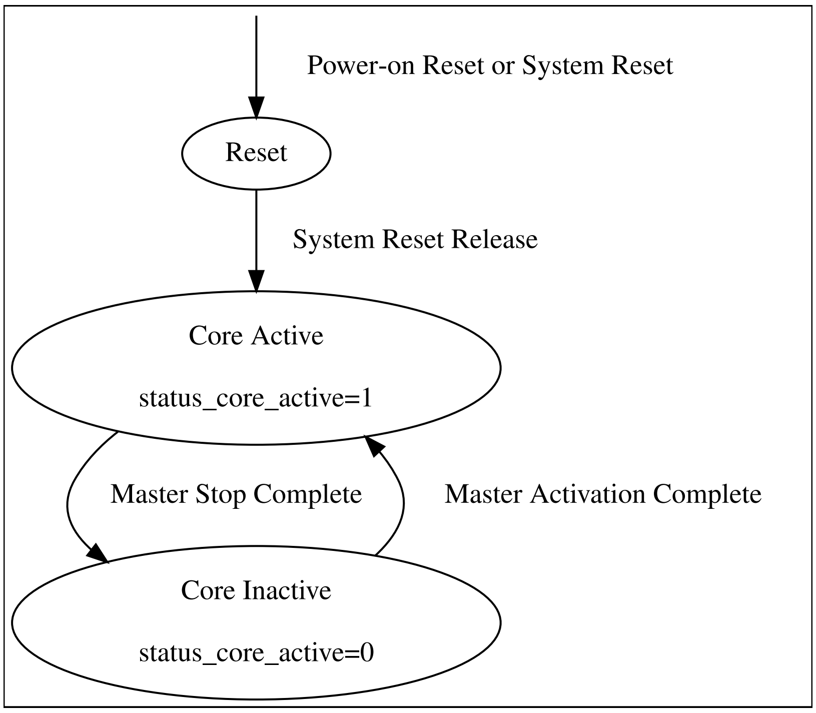


Figure 5. Core-level states transition diagram

Reset

The reset state of a core. No functions of the core can be used.

Core Active

The active state of a core. All functions of the core can be used.

Core Inactive

The inactive state of a core. Program execution functions of the hart in the core are stopped, and the bus master function is also stopped.
If there is a state such as Master Stop where resources in NS31A can be accessed, it is possible to access a register in a core.

The following shows conditions for each state transition.

Power-on Reset or System Reset

Perform the reset according to the procedures described in Section 5.2.1 and Section 5.2.2.

System Reset Release

Release the reset according to the procedure described in Section 5.2.2.

Master Stop Complete

The transition to this state is made when the operation of the hart in the core has been stopped via ncc_mststop_req.

Master Activation Complete

The transition to this state is made when the operation of the hart in the core has been resumed via ncc_active_req.

3.3.3. Hart-level states

Hart-level states are the states of the hart included in the core.
These states indicate execution states, which are independent from the privilege mode of the hart.
Hart-level states are controlled according to state transition requests from the outside of NS31A or CMU, debug control requests from DBG, the execution of instructions, etc.

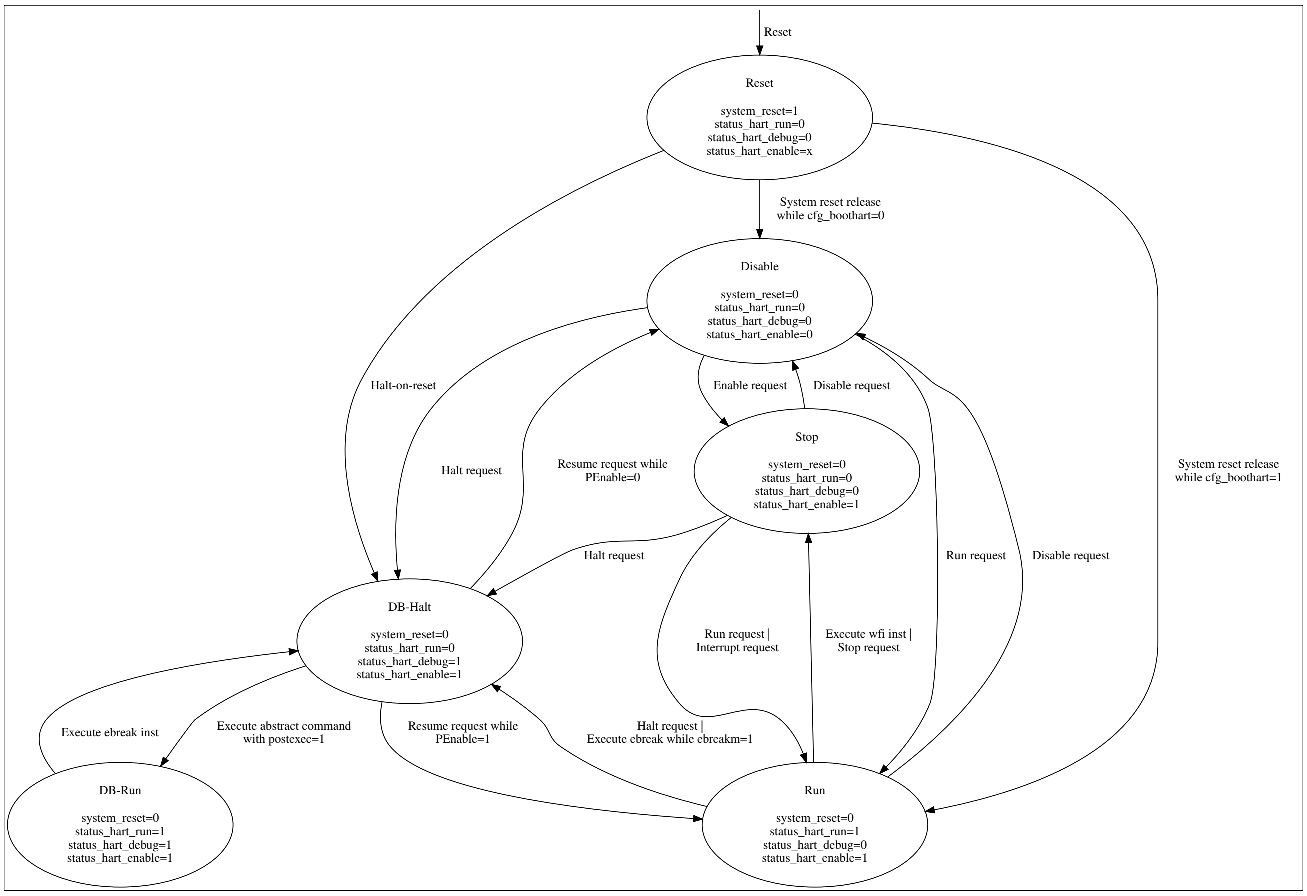


Figure 6. Hart-level state transition diagram

Reset

This is the initial state entered when Power-on Reset or System Reset is performed.

Stop

The state where the execution of a program is stopped. The transition to the Run state is made when an interrupt is acknowledged.

Run

The state where a program is in execution.

Disable

The state where the hart is disabled. The execution of a program is stopped, and interrupt requests are ignored.

DB-Halt

This is one of the states in the Debug Mode. In this state, the execution of a program is temporarily stopped for debugging.

DB-Run

This is one of the states in the Debug Mode. In this state, a debug program is being executed.

> - In a state other than Reset, it is possible to access the memory-mapped register of the hart, regardless of whether the program is running or stopped.
> - If any exception shown in Table 16 or Table 17 (excluding the reset and interrupts) occurs while the Abstract command is executed (DB-Run), the following behavior occurs.
>   - 0x3 is set in the cmderr bit of the abstractcs register.
>   - Branching to the exception vector address is not performed, and the exception-related CSRs (mcause, mepc, mtval, mstatus, mncause, mnepc, and mnstatus) and the exception-related memory mapped register (meecaddr) are not updated.
>   - Transition to the DB-Halt state is made without executing the subsequent instruction of the debug program.

The following describes each state transition in detail.

System reset release

Release the reset according to the procedure described in Section 5.2.2.
The destination state of a state transition changes according to the setting of the external pin cfg_boothart. However, if a Halt request operation is performed during System Reset, the transition to the DB-Halt state, not a normal execution state, is made as a Halt-on-reset.

Halt-on-reset

Request Halt during System Reset by using haltreq or Interlock Halt, and then release the reset according to the procedure described in Section 5.2.2.
In PEnable of mhtenable, the value of cfg_boothart before transition is saved.

Run request

Assert ncc_runhart, and set 1 in RUN of mhtstatus or 1 in HARTRUN.

Stop request

Set 0 in RUN of mhtstatus.

Disable request

Set 0 in Enable of mhtenable or 1 in HARTDIS.

Enable request

Set 1 in Enable of mhtenable.

Halt request

Request Halt by using haltreq of the Debug Module Control Register or Interlock Halt.
At this time, Enable of mhtenable becomes 1, and in PEnable of mhtenable, the value of Enable of mhtenable before transition is saved.

Resume request

Request Resume by using resumereq of the Debug Module Control Register or Interlock Halt.
At this time, the destination state of a state transition changes according to the value of PEnable of mhtenable.

Interrupt request

Generate an interrupt request described in Section 3.5. At this time, the state transition occurs regardless of the value of mstatus.MIE.
Note, however, that the state transition does not occur if the interrupt is disabled by mie. In addition, for the External Interrupt, the state transition occurs only if the corresponding interrupt is enabled in eihenN and the priority of the requested interrupt is higher than the priority that is set in eihth.

Execute wfi inst

Execute the wfi instruction.

Execute ebreak

Execute the ebreak instruction. When the state is Run, the state transition occurs only if ebreakm of dcsr is 1 in M-mode or ebreaku of dcsr is 1 in U-mode.

Execute abstract command with postexec=1

Set 1 in postexec of the Abstract Command Register, and execute the Abstract Command.

Even when the hart is in the DB-Run state, a System-level state can transition to Master Stop or System Stop. In this case, the execution of the debug program is stopped.

3.4. Exceptions

This section describes exceptions supported by NS31A.
An exception is an event that forces the running program to branch to another program. An interrupt is handled as a type of exception. In NS31A, a program that branches due to an exception is run in the M-mode.
In addition to exceptions defined in the RISC-V Priv. Arch. specifications (see Reference Documents for the revision no.), NS31A supports some unique exceptions that are defined according to hardware specifications.

In privilege modes defined in RISC-V, NS31A supports only the Machine mode and User mode. Therefore, exceptions do not occur in the Supervisor mode.
NS31A does not support Page-Based Virtual-Memory System defined in RISC-V. Therefore, the Page fault exception does not occur.

Table 16 and Table 17 show the list of exceptions supported by NS31A.

Table 16. Exception List

Exception Code (mcause)		Description	Type	Vec	mepc	mtval	mstatus			meecaddr
31	30:0						mpie	mie	mpp	
0	0000_0000h	Reset	-	resetvec	Unknown	Unknown	0	0	0	0h
0	0000_0000h	Instruction address misaligned[*1]	Exception	mtvec	currentPC	Access address	current mie	0	*3	s[*2]
0	0000_0001h	Instruction access fault	Exception	mtvec	currentPC	Fetch address	current mie	0	*3	s[*2]
0	0000_0002h	Illegal instruction	Exception	mtvec	currentPC	Instruction Code	current mie	0	*3	s[*2]
0	0000_0003h	Breakpoint	Exception	mtvec	currentPC	0h	current mie	0	*3	s[*2]
0	0000_0004h	Load address misaligned	Exception	mtvec	currentPC	Access address	current mie	0	*3	s[*2]
0	0000_0005h	Load access fault	Exception	mtvec	currentPC	Access address	current mie	0	*3	s[*2]
0	0000_0006h	Store/AMO address misaligned	Exception	mtvec	currentPC	Access address	current mie	0	*3	s[*2]
0	0000_0007h	Store/AMO access fault	Exception	mtvec	currentPC	Access address	current mie	0	*3	s[*2]
0	0000_0008h	Environment call from U-mode	Exception	mtvec	currentPC	0h	current mie	0	0	s[*2]
0	0000_000Bh	Environment call from M-mode	Exception	mtvec	currentPC	0h	current mie	0	3	s[*2]
1	0000_0003h	Software interrupt	Interrupt	mtvec[0]=0: mtvec mtvec[0]=1: mtvec[*4] + 0Ch	nextPC	0h	current mie	0	*3	s[*2]
1	0000_0007h	Timer interrupt	Interrupt	mtvec[0]=0: mtvec mtvec[0]=1: mtvec[*4] + 1Ch	nextPC	0h	current mie	0	*3	s[*2]
1	0000_000Bh	External interrupt	Interrupt	mtvec[0]=0: mtvec mtvec[0]=1: mtvec[*4] + 2Ch	nextPC	0h	current mie	0	*3	s[*2]
1	0000_0013h	Error interrupt	Interrupt	mtvec[0]=0: mtvec mtvec[0]=1: mtvec[*4] + 4Ch	nextPC	0h	current mie	0	*3	s[*2]

*1 When P_RV32C is 1, Instruction address misaligned exceptions do not occur.

*2 Values do not change before and after exceptions.
*3 Depends on the mode in which the exception occurs. (M-mode: 3, U-mode: 0)
*4 When mtvec[0] is 1 (Vectored mode), mtvec[6:2] is ignored. For details of the registers, see Section 3.7.2.2.4.

Table 17. Exception List (NMI)

Exception Code (mncause)		Description	Type	Vec	mnepc	mtval	mnstatus		meecaddr
31	30:0						nmie	mnpp	
1	0000_0002h	External NMI	NMI	nmivec[*1]	nextPC	s[*2]	0	*3	s[*2]
1	0000_0003h	Error NMI	NMI	nmivec[*1]	nextPC	s[*2]	0	*3	s[*2]
1	0000_0003h	Fetch Bus error NMI	NMI	nmivec[*1]	nextPC	s[*2]	0	*3	s[*2]
1	0000_0003h	Non-blocking store/AMO bus error NMI	NMI	nmivec[*1]	nextPC	s[*2]	0	*3	Access address
1	0000_0003h	Non-blocking load bus error NMI	NMI	nmivec[*1]	nextPC	s[*2]	0	*3	Access address

*1 Set by using the INITNMIVEC register of the CMU. For details of the registers, see Section 6.2.
*2 Values do not change before and after exceptions.
*3 Depends on the mode in which the exception occurs. (M-mode: 3, U-mode:0)

In NS31A, for an exception due to the RV32F (access to the floating-point register) instruction, an exception of a type similar to another instruction type (such as RV32I) occurs.

3.4.1. NS31A unique exception list

The following are exceptions that are uniquely defined for NS31A.

- External NMI
  NMI that is generated when the external input signal int_extnmi is asserted.
  The detection method is edge.
- Error NMI
  NMI notified as the Error NMI Request directly from the outside of NS31A via the int_errnmi pin.
  The detection method is edge.
- Error Interrupt
  Interrupt notified as the Error Interrupt Request directly from the outside of NS31A via the int_errint pin.
  The detection method is Level.
- Fetch Bus error NMI
  NMI notified if a fatal error occurs due to instruction fetch.
  Fatal errors include ECC 2-bit errors and multi-hit errors that occur due to an instruction cache, in addition to bus transaction response errors.
- Non-blocking Store/AMO Bus error NMI
  NMI notified if a fatal error occurs when the RV32I Store, RV32C Store, RV32F Store, or RV32A AMO/SC instruction is executed.
  In the AXI configuration, fatal errors include ECC 2-bit errors that occur in local memory, in addition to bus transaction response errors.
- Non-blocking Load Bus error NMI
  NMI notified if a fatal error occurs when the RV32I Load, RV32C Load, RV32F Load, or RV32A LR instruction is executed.
  In the AXI configuration, fatal errors include ECC 2-bit errors that occur in local memory, in addition to bus transaction response errors.

Of the exceptions caused by Load/Store/AMO instruction execution in NS31A, Non-blocking NMI occurs asynchronously.
Due to this, an exception other than the Non-blocking NMI may be detected before a Non-blocking NMI that occurred earlier is detected.
In this case, the address stored in mnepc and the mode stored in mnstatus MNPP due to the occurrence of a Non-blocking NMI are not the address and mode of the instruction that caused the NMI to occur but the value of nextPC and the mode at the instant the NMI occurrence was detected.

Local memory and instruction cache ECC 2-bit errors and instruction cache multi-hit errors are usually classified as Hardware Errors, but in NS31A, they are handled as NMIs like bus errors, and the same code (80000003h) is allocated as the value of mncause when they occur.

3.4.2. Exception priority

Table 18 lists the priority of exceptions in descending order.

Table 18. Exception Priority

m[n]cause	Category	Description
0000_0000h	Async	Reset
8000_0002h[*1]	Async NMI	External NMI
8000_0003h[*1]		Error NMI
8000_0003h[*1]		Non-blocking store bus error NMI
8000_0003h[*1]		Non-blocking load bus error NMI
8000_0013h	Async Interrupt	Error Interrupt
8000_000Bh		External Interrupt
8000_0003h		Software Interrupt
8000_0007h		Timer Interrupt
0000_0003h	Instruction Fetch	Break (Inst. address)
0000_0001h		Instruction access fault (PMP error)
8000_0003h[*1]	Async NMI	Fetch Bus error NMI
0000_0002h	Instruction Decode	Illegal instruction (Reserved inst.)
0000_0002h	Instruction Execution	Illegal instruction (Priv. error)
0000_0002h		Illegal instruction (CSR decode error)
0000_0000h		Instruction address misaligned
0000_000Bh	Environment	Environment call from M-mode
0000_0008h		Environment call from U-mode
0000_0003h		Environment Break (ebreak)
0000_0003h	Load/Store	Break (Load/Store address)
0000_0006h		Store/AMO address misaligned
0000_0004h		Load address misaligned
0000_0007h		Store/AMO access fault (PMP error)
0000_0005h		Load access fault (PMP error)

*1 Code written to mncause.

3.5. Interrupts

This section describes the interrupt processing in NS31A.

3.5.1. Features

NS31A can execute the interrupt processing according to various interrupt requests.
Of interrupt requests, Software/Timer interrupt and External interrupt are requested to the hart via the ACLINT (Advanced Core Local Interruptor) and via the PLIC (Platform-Level Interrupt Controller), respectively, and are executed in M-mode. For details of the ACLINT, see Chapter 7.
For details of the PLIC, see Chapter 8.

Table 19 shows the list of interrupts.

Table 19. NS31A Interrupts

Exception Code	Description	Source
0x03	Software interrupt	ACLINT
0x07	Timer interrupt	ACLINT
0x0B	External interrupt	External
0x13	Error interrupt	External
-	NMI	Various units

The following shows the priority of interrupt requests in descending order.

High priority	Non-maskable interrupt (NMI)
↓	Error interrupt
↓	External interrupt
↓	Software interrupt
Low priority	Timer interrupt

For the priority, including exceptions and NMI, see Table 18.

3.5.2. Software interrupt

Software interrupts are interrupts that are requested when software operates registers in the ACLINT.

For details, see Chapter 7.

3.5.3. Timer interrupt

The timer has a single Timer interrupt notification signal, and the ACLINT notifies the hart of this signal. The hart that will process a Timer interrupt is specified by using the Interrupt Pending bit of the hart.

For details, see Chapter 7.

3.5.4. External interrupt

NS31A has the same number of inputs as that specified in parameter "P_EXTINT_WIDTH". If the priority of the requested External interrupt is higher than the current priority set in the PLIC, this interrupt request is reported to the hart according to the PLIC settings.

For details, see Chapter 8.

3.5.5. Error Interrupt

The Error Interrupt is an interrupt requested by the int_errint pin from the outside of NS31A.

3.5.6. Non-maskable interrupt (NMI)

The Non-maskable interrupt (NMI) is an interrupt requested based on events such as an input from the outside of NS31A. NS31A has the External NMI input and Error NMI input and requests an NMI by using these external input signals. An NMI request is also generated when a Hardware Error or a bus error occurs in NS31A.
For the NMI requests and the registers updated upon NMI acknowledgment, see Table 16.

The NMI controlls the following interrupt requests.

- External NMI
- Error NMI
- Hardware Error NMI
- Bus error NMI

> Hardware Error NMI generation factors include ECC 2-bit errors in local memory and instruction cache, and instruction cache multi-hit errors.
> In NS31A, they are handled as NMIs like bus errors.

For the NMI control handled by NS31A, Resumable NMI in Smrnmi extension is adopted. Smrnmi extension Version 0.4 is adopted. The values of mnstatus, mncause, and mnepc are updated by Resumable NMI.

An NMI clears mnstatus.nmie from 1 to 0 at the time of branching to the handler to suppress interrupts in the handler. If mnstatus.nmie is 0, all exceptions are masked.
To enable an NMI, the software must always set mnstatus.nmie to 1 before use. Once mnstatus.nmie is set to 1 via the software, it cannot be cleared to 0.
Use mnret (0x70200073) to return from the handler. Upon return from the handler, mnepc is restored to the running PC, and mnstatus.nmie is restored from 0 to 1.

When an NMI is acknowledged, it is ignored until the hart branches to the address set by the INITNMIVEC register, even if another NMI request with the same type is issued.
Another NMI can be acknowledged after branching, but in the NMI handler, it is not allowed to skip to another handler while mnstatus.nmie is 0.

3.6. Privileged ISA

This section describes the privilege levels (modes) supported by NS31A.

In RISC-V ISA, a privilege level works as an operating mode of the hardware thread (hart), and the hart operates in its specific privilege level.
Each privilege level has a control status register (CSR) corresponding to the level and provides protection between different software components.

Of the modes defined in RISC-V, NS31A supports the three shown in Table 20.

Table 20. Supported Privilege Mode

Name	Abbreviation
Machine	M
User	U
Debug	D

According to these specifications, those registers and functions that belong to modes other than those described above (e.g., Supervisor mode (S-mode)) are not implemented in NS31A. The Trap Delegation register and function are not implemented either.

3.7. Control and Status Registers

This section describes the control status register (CSR) included in NS31A.
CSR is a register for control and status acquisition, which is mapped to a unique address space, and dedicated access instructions are provided.
NS31A CSRs consists of registers defined in RISC-V Privileged Architectures.

For the CSR supported by NS31A, see Section 3.7.2.

3.7.1. CSR Address Map

Table 21 shows the CSR address space supported by NS31A.

Of the privilege modes defined in RISC-V, only the Machine mode and User mode are supported by NS31A. The Supervisor and Hypervisor modes are not supported.
Therefore, of the CSRs defined for RISC-V, some CSRs are not supported by NS31A. Each register enables access according to Privilege level as indicated in Table 21.

Table 21. NS31A supported CSR Address Map

CSR Address			**Hex**	**Privilege**			**Notes**
[11:10]	**[9:8]**	**[7:6]**		**Debug**	**Machine**	**User**	
00	00	XX	0x000-0x0FF	RW	RW	RW	User "F" CSRs
00	11	XX	0x300-0x3FF	RW	RW	-	Machine Trap Setup / Counter Setup / Trap Handling / Configuration
01	11	01	0x740-0x74F	RW	RW	-	Machine Non-Maskable Interrupt Handling
01	11	10	0x7A0-0x7AF	RW	RW	-	Debug/Trace Registers
01	11	10	0x7B0-0x7BF	RW	-	-	Debug Mode Registers
10	11	00-10	0xB00-0xBBF	RW	RW	-	Machine Counter / Timers
11	00	00-10	0xC00-0xCBF	RO	RO	RO*1	User CSRs
11	11	00-10	0xF00-0xFBF	RO	RO	-	Machine Information Registers

*1 Access in User mode must be enabled by mcounteren.

The CSR address is an address specified in bits [31:20] of a CSR instruction code in RISC-V.
If access is made with a CSR instruction under any of the following conditions, an Illegal instruction exception occurs.

- Access to an unsupported CSR address
  (Some CSRs may be supported or not depending on the configuration parameter specification.)
- Writing to a Read Only CSR
- Access in privilege mode that does not support access

3.7.2. NS31A CSRs

Table 22 shows the list of CSRs supported by NS31A.
For details of CSR specifications, see the Section described in the "Register Descriptions" field.

If the "Register Descriptions" field is blank, see RISC-V Priv. Arch. specifications or RISC-V Debug Support specifications.

Table 22. NS31A CSR Listing

Number	Name	Type	Function	Register Descriptions
User Floating-Point CSRs				
0x001	fflags	RW	Floating-Point Accrued Exceptions	see Section 3.7.2.1.1
0x002	frm	RW	Floating-Point Dynamic Rounding Mode	see Section 3.7.2.1.2
0x003	fcsr	RW	Floating-Point Control and Status Register	see Section 3.7.2.1.3
Machine Trap Setup				
0x300	mstatus	RW	Machine status register.	see Section 3.7.2.2.1
0x301	misa	RW [*1]	ISA and extensions	see Section 3.7.2.2.2
0x304	mie	RW	Machine interrupt-enable register.	see Section 3.7.2.2.3
0x305	mtvec	RW	Machine trap-handler base address.	see Section 3.7.2.2.4
0x306	mcounteren	RW	Machine counter enable	see Section 3.7.2.2.5
0x310	mstatush	RW [*1]	Upper 32bit of Machine status register.	see Section 3.7.2.2.6
Machine Configuration				
0x30A	menvcfg	RW [*1]	Machine environment configuration register.	see Section 3.7.2.3.1
0x31A	menvcfgh	RW [*1]	Upper 32 bits of menvcfg, RV32 only.	see Section 3.7.2.3.2
Machine Counter Setup				
0x320	mcountinhibit	RW	Machine counter-inhibit register.	see Section 3.7.2.4.1
0x323	mhpmevent3	RW	Machine performance-monitoring event selector.	see Section 3.7.2.4.2
…				-
0x32A	mhpmevent10	RW	Machine performance-monitoring event selector.	see Section 3.7.2.4.2
Machine Trap Handling				
0x340	mscratch	RW	Scratch register for machine trap handlers.	see Section 3.7.2.5.1
0x341	mepc	RW	Machine exception program counter.	see Section 3.7.2.5.2
0x342	mcause	RW	Machine trap cause.	see Section 3.7.2.5.3
0x343	mtval	RW	Machine bad address or instruction.	see Section 3.7.2.5.4
0x344	mip	RW [*1]	Machine interrupt pending.	see Section 3.7.2.5.5
Machine Non-Maskable Interrupt Handling				
0x740	mnscratch	RW	Scratch register for machine NMI trap handlers.	see Section 3.7.2.6.1
0x741	mnepc	RW	Machine NMI exception program counter.	see Section 3.7.2.6.2
0x742	mncause	RW	Machine NMI trap cause.	see Section 3.7.2.6.3
0x744	mnstatus	RW	Machine NMI status register.	see Section 3.7.2.6.4
Machine Memory Protection				
0x3A0	pmpcfg0	RW	Physical memory protection configuration.	see Section 3.7.2.7.1
…				-
0x3A3	pmpcfg3	RW	Physical memory protection configuration, RV32 only.	see Section 3.7.2.7.1
0x3B0	pmpaddr0	RW	Physical memory protection address register.	see Section 3.7.2.7.2
…				-
0x3BF	pmpaddr15	RW	Physical memory protection address register.	see Section 3.7.2.7.2
Debug/Trace Registers (shared with Debug Mode)				

Number	Name	Type	Function	Register Descriptions
0x7A0	tselect	RW	Debug/Trace trigger register select.	see Section 3.7.2.8.1
0x7A1	tdata1	RW	First Debug/Trace trigger data register.	see Section 3.7.2.8.2
0x7A2	tdata2	RW	Second Debug/Trace trigger data register.	see Section 3.7.2.8.3
0x7A4	tinfo	RW *1	Trigger Info register.	see Section 3.7.2.8.4
Debug Mode Registers				
0x7B0	dcsr	RW	Debug control and status register.	see Section 3.7.2.9.1
0x7B1	dpc	RW	Debug PC.	see Section 3.7.2.9.2
0x7B2	dscratch0	RW	Debug scratch register 0.	see Section 3.7.2.9.3
Machine Counter/Timers				
0xB00	mcycle	RW	Machine cycle counter.	see Section 3.7.2.10.1
0xB02	minstret	RW	Machine instructions-retired counter.	see Section 3.7.2.10.2
0xB03	mhpmcounter3	RW	Machine Performance-monitoring counter.	see Section 3.7.2.10.3
…				-
0xB0A	mhpmcounter10	RW	Machine Performance-monitoring counter.	see Section 3.7.2.10.3
0xB80	mcycleh	RW	Upper 32 bits of mcycle, RV32 only.	see Section 3.7.2.10.4
0xB82	minstreth	RW	Upper 32 bits of minstret, RV32 only.	see Section 3.7.2.10.5
0xB83	mhpmcounter3h	RW	Upper 32 bits of mhpmcounter3, RV32I only.	see Section 3.7.2.10.6
…				-
0xB8A	mhpmcounter10h	RW	Upper 32 bits of mhpmcounter10, RV32I only.	see Section 3.7.2.10.6
User Counter/Timers				
0xC00	cycle	RO	Cycle counter for RDCYCLE instruction.	see Section 3.7.2.11.1
0xC01	time	RO	Timer for RDTIME instruction.	see Section 3.7.2.11.2
0xC02	instret	RO	Instructions-retired counter for RDINSTRET instruction.	see Section 3.7.2.11.3
0xC03	hpmcounter3	RO	Performance-monitoring counter.	see Section 3.7.2.11.4
…				-
0xC0A	hpmcounter10	RO	Performance-monitoring counter.	see Section 3.7.2.11.4
0xC80	cycleh	RO	Upper 32 bits of cycle, RV32 only.	see Section 3.7.2.11.5
0xC81	timeh	RO	Upper 32 bits of time, RV32 only.	see Section 3.7.2.11.6
0xC82	instreth	RO	Upper 32 bits of instret, RV32 only.	see Section 3.7.2.11.7
0xC83	hpmcounter3h	RO	Upper 32 bits of hpmcounter3, RV32I only.	see Section 3.7.2.11.8
…				-
0xC8A	hpmcounter10h	RO	Upper 32 bits of hpmcounter10, RV32I only.	see Section 3.7.2.11.8
Machine Information Registers				
0xF11	mvendorid	RO	Vendor ID.	see Section 3.7.2.12.1
0xF12	marchid	RO	Architecture ID.	see Section 3.7.2.12.2
0xF13	mimpid	RO	Implementation ID.	see Section 3.7.2.12.3
0xF14	mhartid	RO	Hardware thread ID.	see Section 3.7.2.12.4
0xF15	mconfigptr	RO	Pointer to configuration data structure.	see Section 3.7.2.12.5

*1 All the bit fields of misa, mstatush, menvcfg, menvcfgh, mip, and tinfo are read-only; however, these registers are defined as readable and writable registers in RISC-V architecture specifications. Any write access to these registers does not cause illegal instruction exceptions or bus errors.

3.7.2.1. User Floating-Point CSRs

3.7.2.1.1. fflags

fflags is a register that holds the flag of an exception that occurred during floating-point calculation.
It is a CSR defined in RISC-V Priv. Arch.
The entity of the fflags field in this register is the same as the FFLAGS field of the fcsr register.
This register can be read/written when configuration parameter P_FPU is 1 and the FS field of mstatus is 11. An illegal instruction exception occurs if this register is accessed in other cases.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–			fflags		
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW

Table 23. Floating-Point Accrued Exceptions (fflags)

Bits	Field Name	Type	Reset	Description
[31:5]	-	RO	0x0	Reserved
[4:0]	fflags	RW	0x0	Floating-Point Accrued Exception Flags

3.7.2.1.2. frm

frm is a register for setting the rounding mode of floating-point calculation.
It is a CSR defined in RISC-V Priv. Arch.
The entity of the frm field in this register is the same as the FRM field of the fcsr register.
This register can be read/written when configuration parameter P_FPU is 1 and the FS field of mstatus is 11. An illegal instruction exception occurs if this register is accessed in other cases.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–		frm	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW

Table 24. Floating-Point Rounding Mode (frm)

Bits	Field Name	Type	Reset	Description
[31:3]	-	RO	0x0	Reserved
[2:0]	frm	RW	0x0	Floating-Point Rounding Mode

3.7.2.1.3. fcsr

fcsr is a register for controlling floating-point calculation and acquiring its status.

It is a CSR defined in RISC-V Priv. Arch.
This register can be read/written when configuration parameter P_FPU is 1 and the FS field of mstatus is 11. An illegal instruction exception occurs if this register is accessed in other cases.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–		FRM				FFLAGS		
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW

Table 25. FPU CSR (FCSR)

Bits	Field Name	Type	Reset	Description
[31:8]	-	RO	0x0	Reserved
[7:5]	FRM	RW	0x0	Floating-Point Dynamic Rounding Mode 000: rne/ round-to-nearest, ties to even 001: rtz/ round-to-towards zero 010: rdn/ round-down (towards -∞) 100: rmm/ round-to-nearest, ties to max magnitude 011: rup/ round-up (towards +∞) 101, 110, 111: Invalid
[4:0]	FFLAGS	RW	0	Floating-Point Accrued Exceptions The accrued exception flags indicate the exception conditions that have arisen on any floating-point arithmetic instruction since the field was last reset by software, as shown below. bit4: NV/ Invalid Operation bit3: DZ/ Divide by Zero bit2: OF/ Overflow bit1: UF/ Underflow bit0: NX/ Inexac

3.7.2.2. Machine Trap Setup

3.7.2.2.1. mstatus

mstatus is a CSR for acquiring and controlling the current operating status of the hart.
It is a CSR defined in RISC-V Priv. Arch.
NS31A supports only the functions related to the Machine mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	SD	—	—	—	—	—	—	—	—	—	TW	—	—	—	MPRV	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO	RO	RO	RW	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	FS[1:0]		MPP[1:0]		—	—	—	MPIE	—	—	—	MIE	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RW	RW	RW	RW	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO

Table 26. Machine Status Register (mstatus)

Bits	Field Name	Type	Reset	Description
[31]	SD	RO	0x0	Summary of whether FS field is dirty 1’b0: The FS field is 2’b00 (Off). 1’b1: The FS field is 2’b11 (Dirty). If the FPU is not implemented (P_FPU = 0), always 0.
[30:22]	-	RO	0x0	Reserved
[21]	TW	RW	0x0	Timeout wait for intercepting the WFI instruction 1’b0: By executing the WFI instruction in U-mode, the hart-level status is changed to Stop. 1’b1: By executing the WFI instruction in U-mode, an Illegal Instruction exception occurs.
[20:18]	-	RO	0x0	Reserved
[17]	MPRV	RW	0x0	Modify PRiVilege 1’b0 : For Load/Store in M-mode, a normal access privilege is granted. 1’b1 : For Load/Store in M-mode, the access privilege in the mode that is set in MPP is granted. They are cleared to "0" upon return from M-mode/D-mode to U-mode.
[16:15]	-	RO	0x0	Reserved
[14:13]	FS[1:0]	RW	2’b00	Floating-point context status for Machine mode 2’b00 : Off 2’b01 : Reserved 2’b10 : Reserved 2’b11 : Dirty To enable the execution of floating-point instructions, it is needed to set this field to a value other than 2’b00. If a floating-point instruction is executed when this field is 2’b00, an illegal instruction exception will be thrown. If a value other than 2’b00 is written, 2’b11 (Dirty) will be set. If the FPU is not implemented (P_FPU = 0), RO (which is always 0).

Bits	Field Name	Type	Reset	Description
[12:11]	MPP[1:0]	RW	2’b00	Previous privilege mode for Machine mode This field holds the Privilege level when a Trap occurred. 2’b00 : User mode 2’b01 : Reserved 2’b10 : Reserved 2’b11 : Machine mode If a Reserved value is written, it is ignored. Restored as a Privilege level when the state is returned to the previous state before the occurrence of Trap due to the execution of the mret instruction. After the execution of the mret instruction, this field is cleared to "0".
[10:8]	-	RO	0x0	Reserved
[7]	MPIE	RW	0x0	previous interrupt-enable bit for Machine mode This field holds the interrupt enable bit (mstatus.MIE) when a trap occurred. "1" is set when the state is returned to the previous state before the occurrence of a trap due to the mret instruction.
[6:4]	-	RO	0x0	Reserved
[3]	MIE	RW	0x0	global interrupt-enable bit for Machine mode This field is cleared to "0" when an interrupt or exception is acknowledged. The value held in mstatus.MPIE is restored in this field when the state is returned to the previous state before the occurrence of a trap due to the mret instruction.
[2:0]	-	RO	0x0	Reserved

If MPRV is 1, the following operations caused by Load/Store in M-mode are performed with the privilege of the mode that is set in MPP.

- Access protection with PMP
- Enabling access to the Internal Register and the Hart Context Register
- Value of AxPROT[0] output from the System Bus Master

In U-mode, MPRV does not affect the above operations; they are always performed with the privilege equivalent to U-mode. In D-mode, MPRV does not affect them because dcsr.mprven is not implemented in NS31A.
When mnstatus.NMIE is 0, MPRV does not affect them because MPRV is handled as being 0.

In U-mode, M-mode Interrupts are enabled regardless of the value of MIE.
In NS31A, all interrupts are handled as M-mode Interrupts.

If the mret instruction is executed in the Debug mode, each bit field of mstatus is updated as in the non-Debug mode while the privilege mode is not updated and remains the Debug mode.

3.7.2.2.2. misa

misa is a CSR for acquiring ISA information supported by the hart.
It is a CSR defined in RISC-V Priv. Arch.
In NS31A, the MXL field is fixed to a value indicating 32 bits.
The Extensions field is fixed to a value indicating the RISC-V extended specifications supported by NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	MXL		—	—	—	—	Extensions									
Reset:	0	1	0	0	0	0	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	Extensions															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) refer to Description of "Externsions" Field

Table 27. Machine ISA Register (misa)

Bits	Field Name	Type	Reset	Description
[31:30]	MXL	RO	0x1	Machine XLEN (width of an integer width) 0x1: 32bit
[29:26]	-	RO	0x0	Reserved

[25:0]	Extensions	RO	See the Description column.	Indicates the supported RISC-V extended specifications.
				Bit25=0 : (Z) Reserved
				Bit24=0 : (Y) Reserved
				Bit23=0 : (X) Non-standard extensions present
				Bit22=0 : (W) Reserved
				Bit21=0 : (V) Vector extension
				Bit20=1 : (U) User mode implemented
				Bit19=0 : (T) Transactional Memory extension
				Bit18=0 : (S) Supervisor mode implemented
				Bit17=0 : (R) Reserved
				Bit16=0 : (Q) Quad-precision floating-point extension
				Bit15=0 : (P) Packed-SIMD extension
				Bit14=0 : (O) Reserved
				Bit13=0 : (N) User-level interrupts supported
				Bit12=1 : (M) Integer Multiply/Divide extension
				Bit11=0 : (L) Decimal Floating-Point extension
				Bit10=0 : (K) Reserved
				Bit9=0 : (J) Dynamically Translated Languages extension
				Bit8=0 (P_RV32E=1), 1 (P_RV32E=0) : (I) RV32I base ISA
				Bit7=0 : (H) Hypervisor extension
				Bit6=0 : (G) Additional standard extension present
				Bit5=0 (P_FPU=0), 1 (P_FPU=1) : (F) Single-precision floating-point extension
				Bit4=0 (P_RV32E=0), 1 (P_RV32E=1) : (E) RV32E base ISA
				Bit3=0 : (D) Double-precision floating-point extension
				Bit2=0 (P_RV32C=0), 1 (P_RV32C=1) : (C) Compressed extension
				Bit1=0 : (B) Tentatively reserved for Bit-Manipulation extension
				Bit0=1 : (A) Atomic extension

3.7.2.2.3. mie

mie is a CSR for controlling the permission of acknowledging each interrupt request in the hart.
It is a CSR defined in RISC-V Priv. Arch.
NS31A supports the functions related to the Machine mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	ERIE	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	MEIE	—	—	—	MTIE	—	—	—	MSIE	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO

Table 28. Machine interrupt-enable register (mie)

Bits	Field Name	Type	Reset	Description
[31:20]	-	RO	0x0	Reserved
[19]	ERIE	RW	0x0	Error Interrupt Enable (a unique specification of NS31A) 1: Enable 0: Disable
[18:12]	-	RO	0x0	Reserved
[11]	MEIE	RW	0x0	Machine External Interrupt Enable 1:Enable 0:Disable
[10:8]	-	RO	0x0	Reserved
[7]	MTIE	RW	0x0	Machine Timer Interrupt Enable 1:Enable 0:Disable
[6:4]	-	RO	0x0	Reserved
[3]	MSIE	RW	0x0	Machine Software Interrupt Enable 1:Enable 0:Disable
[2:0]	-	RO	0x0	Reserved

3.7.2.2.4. mtvec

mtvec is a register for storing the base address of each trap handler excluding Reset and NMI.
It is a CSR defined in RISC-V Priv. Arch.
NS31A supports the Direct and Vectored modes.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	BASE															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	BASE														MODE	
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	0	(*2)
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RW

(*1) Value set to cfg_mtvec[31:2]
(*2) Value set to cfg_mtvec[0]

Table 29. Machine trap-handler base address Register (mtvec)

Bits	Field Name	Type	Reset	Description
[31:2]	BASE	RW	cfg_mtvec[31:2] setting value	vector base address.bit[6:2] is masked with 0 in Vectored mode.
[1:0]	MODE	[1]:RO [0]:RW	[1]:0x0 [0]:cfg_mtvec[0] setting value	vector mode 3 : Reserved 2 : Reserved 1 : Vectored 0 : Direct

In the Direct mode, the vector addresses of all exceptions (including interrupts) are set to the addresses (PC[31:0]={mtvec[31:2],00}) that are aligned with 4-byte boundaries.
In the Vectored mode, the vector address of an interrupt other than NMI is set to the address PC[31:0]={mtvec[31:7],Cause[4:0],00}) obtained by concatenating the value of bits 31-7 of the BASE field and the value of the bits obtained by quadrupling the lower 5 bits of the interrupt factor number (exception code).
The setting value of mtvec[6:2] is ignored.
For the factor number of an interrupt, see Table 19.
In the Vectored mode, the vector address of an exception is also set to the address (PC[31:0]={mtvec[31:2],00}) that is the same as that in the Direct mode.

If the Reserved value (2) is written in the MODE[1:0] field, the MODE field is set to 0 (Direct mode).
If the Reserved value (3) is written in the MODE[1:0] field, the MODE field is set to 1 (Vectored mode).

In the Vectored mode, the base address is aligned to 128 bytes.

3.7.2.2.5. mcounteren

mcounteren is a register for setting whether to enable access to the performance monitor counter in User mode.
It is a CSR defined in RISC-V Priv. Arch.
The fields to be implemented, HPM3 to HPM10, depend on the number of counters specified in configuration parameter P_HPMNUM.
The HPMn bit corresponding to an unsupported counter numbers is RO. 0 is always read.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	HPM10	HPM9	HPM8	HPM7	HPM6	HPM5	HPM4	HPM3	IR	TM	CY
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 30. Machine Counter Enable Register (mcounteren)

Bits	Field Name	Type	Reset	Description
[31:11]	-	RO	0x0	Reserved
[10:3]	HPM3-10	RW	0x0	Setting whether to enable access to hpmcounter3-10 and hpmcounter3h-10h in User mode 1’b0 : Disables access. (An Illegal instruction exception is thrown.) 1’b1 : Enables access.
[2]	IR	RW	0x0	Setting whether to enable access to instret and instreth in User mode 1’b0 : Disables access. (An Illegal instruction exception is thrown.) 1’b1 : Enables access.
[1]	TM	RW	0x0	Setting whether to enable access to time and timeh in User mode 1’b0 : Disables access. (An Illegal instruction exception is thrown.) 1’b1 : Enables access.
[0]	CY	RW	0x0	Setting whether to enable access to cycle and cycleh in User mode 1’b0 : Disables access. (An Illegal instruction exception is thrown.) 1’b1 : Enables access.

3.7.2.2.6. mstatush

mstatush is the upper 32 bits of a register for acquiring and controlling the current operating status of a hart.
It is a CSR defined in RISC-V Priv. Arch.
NS31A supports only the functions related to the Machine mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	MBE	SBE	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 31. Upper 32 bits of Machine Status Register (mstatush)

Bits	Field Name	Type	Reset	Description
[31:6]	-	RO	0x0	Reserved
[5]	MBE	RO	0x0	Machine endianness of memory (0:little, 1:big)
[4]	SBE	RO	0x0	Supervisor endianness of memory (0:little, 1:big)
[3:0]	-	RO	0x0	Reserved

MBE represents endian in Machine mode, and only MBE=0 (little endian) is supported.

3.7.2.3. Machine Configuration

3.7.2.3.1. menvcfg

menvcfg is a register for controlling the behavior of instruction execution in User mode.
It is a CSR defined in RISC-V Priv. Arch.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	CBZE	CBCFE	CBIE		—	—	—	FIOM
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 32. Machine Environment Configuration (menvcfg)

Bits	Field Name	Type	Reset	Description
[31:8]	-	RO	0x0	Reserved
[7]	CBZE	RO	0x0	Cache Block Zero instruction Enable
[6]	CBCFE	RO	0x0	Cache Block Clean and Flush instruction Enable
[5:4]	CBIE	RO	0x0	Cache Block Invalidate instruction Enable 00: The instruction raises an illegal instruction or virtual instruction exception 01: The instruction is executed and performs a flush operation 10: Reserved 11: The instruction is executed and performs an invalidate operation
[3:1]	-	RO	0x0	Reserved
[0]	FIOM	RO	0x0	Fence of I/O implies Memory

3.7.2.3.2. menvcfgh

menvcfgh is a register for controlling Execution Environment from User mode.
menvcfgh holds the upper 32 bits of the register value.
It is a CSR defined in RISC-V Priv. Arch.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	STCE	PBMTE	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 33. Upper 32 bits of menvcfg (menvcfgh)

Bits	Field Name	Type	Reset	Description
[31]	STCE	RO	0x0	Furnished by the forthcoming Sstc extension
[30]	PBMTE	RO	0x0	Svpbmt extension is available for use in S-mode and G-stage address translation
[29:0]	-	RO	0x0	Reserved

3.7.2.4. Machine Counter Setup

3.7.2.4.1. mcountinhibit

mcountinhibit is a CSR for suppressing/controlling the increment of each counter for the observation of hardware performance.
It is a CSR defined in RISC-V Priv. Arch.
NS31A supports the bits related to mcycle, minstret, and mhpmcounter.

The fields to be implemented, HPM3 to HPM10, depend on the number of counters specified in configuration parameter P_HPMNUM.
The HPMn bit corresponding to an unsupported counter numbers is RO. 0 is always read.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	HPM10	HPM9	HPM8	HPM7	HPM6	HPM5	HPM4	HPM3	IR	—	CY
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RW

Table 34. Machine counter-inhibit Register (mcountinhibit)

Bits	Field Name	Type	Reset	Description
[31:11]	-	RO	0x0	Reserved
[10:3]	HPM3-10	RW	0x0	HPM3-10 1: Do not increment the counter for any of mhpmcounter3 to mhpmcounter10. 0: Increment the counter.
[2]	IR	RW	0x0	IR 1: Do not increment the counter for minstret. 0: Increment the counter.
[1]	-	RO	0x0	Reserved
[0]	CY	RW	0x0	CY 1: Do not increment the counter for mcycle. 0: Increment the counter.

The count condition for mcycle is not determined by the above register settings alone.
For details, see Section 3.7.2.10.1.

3.7.2.4.2. mhpmevent3-10

Machine Hardware Performance Monitor events (mhpmevent3 to mhpmevent10) are used to set events for acquiring the execution performance of the HART to be counted with mhpmcounter3 to mhpmcounter10. In each of mhpmevent3 to mhpmevent10, set, for a counter, the Class of the event to be acquired and the Masks for enabling and disabling the event. For details, see Section 3.7.3.
It is a register that can be read/written in the Machine mode. The number of registers that will be actually implemented can be set by setting configuration parameter P_HPMNUM.
Accessing an unimplemented register causes an Illegal Instruction Exception to be thrown.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	mask									
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mask								—	—	—	—	—	—	class	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO	RW	RW

Table 35. Machine performance-monitoring event selector Register (mhpmevent3-10)

Bits	Field Name	Type	Reset	Description
[31:26]	-	RO	0x0	Reserved
[25:8]	mask	RW	0x0	HPM Event Mask 1:enable (When an event corresponding to the Event Mask bit occurs, the counter is incremented.) 0:disable (The counter is not incremented.) (Even when 1 is written to a bit to which an event in Event Class between 0x0 and 0x2, described in Section 3.7.3, is not allocated, the counter is not incremented.)
[7:2]	-	RO	0x0	Reserved
[1:0]	class	RW	0x0	HPM Event Class 3:Reserved 2:Event Class=0x2 selected 1:Event Class=0x1 selected 0:Event Class=0x0 selected

3.7.2.5. Machine Trap Handling

3.7.2.5.1. mscratch

mscratch is a CSR for scratch used in the Machine mode.
Generally, it is used by the trap handler in the Machine mode.
It is a CSR defined in RISC-V Priv. Arch.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mscratch															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mscratch															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x: Undefined

Table 36. Machine Trap handler’s scratch Register (mscratch)

Bits	Field Name	Type	Reset	Description
[31:0]	mscratch	RW	Undefined	scratch register for Machine-mode.

3.7.2.5.2. mepc

mepc is a CSR for holding the PC when an exception (including an interrupt) is acknowledged.
It is used as the return address of the mret instruction.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The return address can be changed by software.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mepc[31:2]															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mepc[31:2]														mepc[1]	—
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO

x: undefined

Table 37. Machine Exception PC Register (mepc)

Bits	Field Name	Type	Reset	Description
[31:2]	mepc[31:2]	RW	Undefined	Exception PC[31:2]
[1]	mepc[1]	RW	Undefined	Exception PC[1] RO when P_RV32C = 0. 0 is always read.
[0]	-	RO	0x0	Reserved

If the mret instruction is executed in the Debug mode, the PC is updated using the value of mepc as in the non-Debug mode while the privilege mode is not changed.

3.7.2.5.3. mcause

mcause is a CSR for holding the factor number (Exception code) when an exception (including an interrupt) of the hart is acknowledged.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The value can be changed by software.
For the factor number of an exception, see Table 16.
For the factor number of an interrupt, see Table 19.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	Interrupt	mcause[30:16]														
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mcause[15:5]											mcause[4:0]				
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW

Table 38. Machine Exception Cause Register (mcause)

Bits	Field Name	Type	Reset	Description
[31]	Interrupt	RW	0x0	Interrupt Bit 1 : trap was caused by an interrupt. 0 : trap was not caused by an interrupt.
[30:5]	mcause[30:5]	RW	0x0	Exception Code (Code is always 0.)

Bits	Field Name	Type	Reset	Description
[4:0]	mcause[4:0]	RW	0x0	Exception Code

3.7.2.5.4. mtval

mtval is a CSR for holding exception-specific information when an exception (including an interrupt) is acknowledged.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The value can be changed by software.
For the value stored when an exception is acknowledged, see Table 16. When an interrupt is acknowledged, “0” is stored.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mtval															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mtval															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x: undefined

Table 39. Machine Trap Value Register (mtval)

Bits	Field Name	Type	Reset	Description
[31:0]	mtval	RW	Undefined	Trap Value

3.7.2.5.5. mip

mip is a CSR for controlling the pending bit of each interrupt request in the hart.
It is a CSR defined in RISC-V Priv. Arch.
NS31A supports the functions related to the Machine mode.
Bit 19 is the pending bit for error interrupts, and it is a unique extension bit of NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	ERIP	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	MEIP	—	—	—	MTIP	—	—	—	MSIP	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 40. Machine interrupt-pending register (mip)

Bits	Field Name	Type	Reset	Description
[31:20]	-	RO	0x0	Reserved
[19]	ERIP	RO	0x0	Error Interrupt Pending Bit (a unique specification of NS31A) 1: Exception is pending 0: Exception is not pending
[18:12]	-	RO	0x0	Reserved
[11]	MEIP	RO	0x0	Machine External Interrupt Pending Bit 1: Exception is pending 0: Exception is not pending
[10:8]	-	RO	0x0	Reserved

Bits	Field Name	Type	Reset	Description
[7]	MTIP	RO	0x0	Machine Timer Interrupt Pending Bit 1: Exception is pending 0: Exception is not pending
[6:4]	-	RO	0x0	Reserved
[3]	MSIP	RO	0x0	Machine Software Interrupt Pending Bit 1: Exception is pending 0: Exception is not pending
[2:0]	-	RO	0x0	Reserved

3.7.2.6. Machine Non-Maskable Interrupt Handling

3.7.2.6.1. mnscratch

mnscratch is a register for scratch used in the Machine mode.
Generally, it is used by the NMI handler in the Machine mode.
It is a CSR defined in RISC-V Priv. Arch.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mnscratch															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mnscratch															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 41. Machine Resumable Non Maskable scratch Register (mnscratch)

Bits	Field Name	Type	Reset	Description
[31: 0]	mnscratch	RW	Undefined	scratch register for Resumable Non Maskable.

3.7.2.6.2. mnepc

mnepc is a register for holding the PC when an NMI is acknowledged.
It is used as the return address of the mnret instruction.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The return address can be changed by software.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mnepc[31:2]															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mnepc[31:2]														mnepc[1]	—
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO

Table 42. Machine Resumable Non Maskable Exception PC Register (mnepc)

Bits	Field Name	Type	Reset	Description
[31:2]	mnepc[31:2]	RW	Undefined	Exception PC[31:2]
[1]	mnepc[1]	RW	Undefined	Exception PC[1] RO when P_RV32C = 0. 0 is always read.
[0]	-	RO	0x0	Reserved

If the mnret instruction is executed in the Debug mode, the PC is updated using the value of mnepc as in the non-Debug mode while the privilege mode is not changed.

3.7.2.6.3. mncause

mncause is a register for holding the factor number (Exception code) when an NMI is acknowledged.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The value can be changed by software.
For the factor number of an NMI, see Table 17.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	Interrupt	mncause[30:16]														
Reset:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mncause[15:2]														mncause[1:0]	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW

Table 43. Machine Resumable Non Maskable Exception Cause Register (mncause)

Bits	Field Name	Type	Reset	Description
[31]	Interrupt	RO	0x1	Interrupt Bit (Code is always 1.)
[30:2]	mncause[30:2]	RO	0x0	Exception Code (Code is always 0.)
[1:0]	mncause[1:0]	RW	0x0	Exception Code

3.7.2.6.4. mnstatus

mnstatus is a register for acquiring and controlling the NMI operating status of a hart.
It is a CSR defined in RISC-V Priv. Arch.
NS31A supports only the functions related to the Machine mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	MNPP		–	–	–	–	–	–	–	NMIE	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RW	RW	RO	RO	RO	RO	RO	RO	RO	RW	RO	RO	RO

Table 44. Machine Resumable Non Maskable Status Register (mnstatus)

Bits	Field Name	Type	Reset	Description
[31:13]	-	RO	0x0	Reserved
[12:11]	MNPP[1:0]	RW	2'b00	Previous privilege mode for Machine mode Maintains the privilege level when a Trap occurs. 2'b00 : User mode 2'b01 : Reserved 2'b10 : Reserved 2'b11 : Machine Mode Ignored when lighting the value of Reserved. When the mnret instruction returns to the state before the Trap occurred, it is restored as Privilege level. It is cleared to "0" after execution of the mnret instruction.
[10:4]	-	RO	0x0	Reserved
[3]	NMIE	RW	0x0	When NMI Enable NMIE is 0, the setting for the mstatus.MPRV bit is invalid. In addition, all interrupts are prohibited. After the execution of the mnret instruction, this field is set to "1".
[2:0]	-	RO	0x0	Reserved

mnstatus.NMIE is reset to 0, so it is necessary to always set it to 1 by software before using an NMI.
Once mnstatus.NMIE is set via software, it will not be cleared.
If the mnret instruction is executed in the Debug mode, each bit field of mnstatus is updated as in the non-Debug mode while the privilege mode is not updated and remains the Debug mode.

3.7.2.7. Machine Memory Protection

3.7.2.7.1. pmpcfg0-3

pmpcfg0-3 are registers for configuring each entry in Physical Memory Protection.
pmpcfg0 stores settings for entries 0-3, pmpcfg1 stores settings for entries 4-7, pmpcfg2 stores settings for entries 8-11, and pmpcfg3 stores settings for entries 12-15.

When n=0,

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	pmp3cfg.L	—	—	pmp3cfg.A		pmp3cfg.X	pmp3cfg.W	pmp3cfg.R	pmp2cfg.L	—	—	pmp2cfg.A		pmp2cfg.X	pmp2cfg.W	pmp2cfg.R
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RO	RO	RW	RW	RW	RW	RW	RW	RO	RO	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	pmp1cfg.L	—	—	pmp1cfg.A		pmp1cfg.X	pmp1cfg.W	pmp1cfg.R	pmp0cfg.L	—	—	pmp0cfg.A		pmp0cfg.X	pmp0cfg.W	pmp0cfg.R
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RO	RO	RW	RW	RW	RW	RW	RW	RO	RO	RW	RW	RW	RW	RW

Table 45. PMP configuration register n (pmpcfgn, n=0-3)

Bits	Field Name		Type	Reset	Description
[31]	n = 0	pmp3cfg.L	RW	0x0	Lock bit 1: Lock 0: UnLock
	n = 1	pmp7cfg.L			
	n = 2	pmp11cfg.L			
	n = 3	pmp15cfg.L			
[30:29]	-		RO	0x0	Reserved
[28:27]	n = 0	pmp3cfg.A	RW	0x0	address-matching mode 0: (OFF) Null region (disable) 1: (TOR) Top of range 2: (NA4) Naturally aligned four-byte region 3: (NAPOT) Naturally aligned power-of-two region, ≧8 bytes, ≦4096 MBytes
	n = 1	pmp7cfg.A			
	n = 2	pmp11cfg.A			
	n = 3	pmp15cfg.A			
[26]	n = 0	pmp3cfg.X	RW	0x0	1: Instruction execution is permitted 0: Instruction execution is not permitted
	n = 1	pmp7cfg.X			
	n = 2	pmp11cfg.X			
	n = 3	pmp15cfg.X			
[25]	n = 0	pmp3cfg.W	RW	0x0	1: Write is permitted 0: Write is not permitted
	n = 1	pmp7cfg.W			
	n = 2	pmp11cfg.W			
	n = 3	pmp15cfg.W			
[24]	n = 0	pmp3cfg.R	RW	0x0	1: Read is permitted 0: Read is not permitted
	n = 1	pmp7cfg.R			
	n = 2	pmp11cfg.R			
	n = 3	pmp15cfg.R			
[23]	n = 0	pmp2cfg.L	RW	0x0	Lock bit 1: Lock 0: UnLock
	n = 1	pmp6cfg.L			
	n = 2	pmp10cfg.L			
	n = 3	pmp14cfg.L			
[22:21]	-		RO	0x0	Reserved

Bits	Field Name		Type	Reset	Description
[20:19]	n = 0	pmp2cfg.A	RW	0x0	address-matching mode 0: (OFF) Null region (disable) 1: (TOR) Top of range 2: (NA4) Naturally aligned four-byte region 3: (NAPOT) Naturally aligned power-of-two region, ≧8 bytes, ≦4096 MBytes
	n = 1	pmp6cfg.A			
	n = 2	pmp10cfg.A			
	n = 3	pmp14cfg.A			
[18]	n = 0	pmp2cfg.X	RW	0x0	1: Instruction execution is permitted 0: Instruction execution is not permitted
	n = 1	pmp6cfg.X			
	n = 2	pmp10cfg.X			
	n = 3	pmp14cfg.X			
[17]	n = 0	pmp2cfg.W	RW	0x0	1: Write is permitted 0: Write is not permitted
	n = 1	pmp6cfg.W			
	n = 2	pmp10cfg.W			
	n = 3	pmp14cfg.W			
[16]	n = 0	pmp2cfg.R	RW	0x0	1: Read is permitted 0: Read is not permitted
	n = 1	pmp6cfg.R			
	n = 2	pmp10cfg.R			
	n = 3	pmp14cfg.R			
[15]	n = 0	pmp1cfg.L	RW	0x0	Lock bit 1: Lock 0: UnLock
	n = 1	pmp5cfg.L			
	n = 2	pmp9cfg.L			
	n = 3	pmp13cfg.L			
[14:13]	-		RO	0x0	Reserved
[12:11]	n = 0	pmp1cfg.A	RW	0x0	address-matching mode 0: (OFF) Null region (disable) 1: (TOR) Top of range 2: (NA4) Naturally aligned four-byte region 3: (NAPOT) Naturally aligned power-of-two region, ≧8 bytes, ≦4096 MBytes
	n = 1	pmp5cfg.A			
	n = 2	pmp9cfg.A			
	n = 3	pmp13cfg.A			
[10]	n = 0	pmp1cfg.X	RW	0x0	1: Instruction execution is permitted 0: Instruction execution is not permitted
	n = 1	pmp5cfg.X			
	n = 2	pmp9cfg.X			
	n = 3	pmp13cfg.X			
[9]	n = 0	pmp1cfg.W	RW	0x0	1: Write is permitted 0: Write is not permitted
	n = 1	pmp5cfg.W			
	n = 2	pmp9cfg.W			
	n = 3	pmp13cfg.W			
[8]	n = 0	pmp1cfg.R	RW	0x0	1: Read is permitted 0: Read is not permitted
	n = 1	pmp5cfg.R			
	n = 2	pmp9cfg.R			
	n = 3	pmp13cfg.R			
[7]	n = 0	pmp0cfg.L	RW	0x0	Lock bit 1: Lock 0: UnLock
	n = 1	pmp4cfg.L			
	n = 2	pmp8cfg.L			
	n = 3	pmp12cfg.L			

Bits	Field Name		Type	Reset	Description
[6:5]	-		RO	0x0	Reserved
[4:3]	n = 0	pmp0cfg.A	RW	0x0	address-matching mode 0: (OFF) Null region (disable) 1: (TOR) Top of range 2: (NA4) Naturally aligned four-byte region 3: (NAPOT) Naturally aligned power-of-two region, ≥8 bytes, ≦4096 MBytes
	n = 1	pmp4cfg.A			
	n = 2	pmp8cfg.A			
	n = 3	pmp12cfg.A			
[2]	n = 0	pmp0cfg.X	RW	0x0	1: Instruction execution is permitted 0: Instruction execution is not permitted
	n = 1	pmp4cfg.X			
	n = 2	pmp8cfg.X			
	n = 3	pmp12cfg.X			
[1]	n = 0	pmp0cfg.W	RW	0x0	1: Write is permitted 0: Write is not permitted
	n = 1	pmp4cfg.W			
	n = 2	pmp8cfg.W			
	n = 3	pmp12cfg.W			
[0]	n = 0	pmp0cfg.R	RW	0x0	1: Read is permitted 0: Read is not permitted
	n = 1	pmp4cfg.R			
	n = 2	pmp8cfg.R			
	n = 3	pmp12cfg.R			

In NS31A, functions are expanded as follows.

- A unique function is added for enabling the write access to the L bit.
- The pmpcfg register (0-3) to be implemented depends on the number of entries specified in configuration parameter P_PMPNUM. If any pmpcfg register corresponding to an entry other than the specified ones is accessed, an illegal instruction exception occurs.

According to the unique specification of the L bit, write to the pmpncfg (n=0-15) field (including the L bit) is valid only in any of the following cases. Otherwise, the write access is invalid. If an invalid write access is performed, no interrupt or exception will occur.

- If the "L" bit of the corresponding field is UnLocked (pmpncfg.L=0)
- If the PMP Config Unlock signal (ncc_pmpcfgul) is High

3.7.2.7.2. pmpaddr0-15

pmpaddr0-15 are CSRs for setting the address ranges of entries 0-15 in Physical Memory Protection.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	address[33:32]		address[31:18]													
Reset:	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	address[17:2]															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x: Undefined

Table 46. PMP address register n (pmpaddrn, n=0~15)

Bits	Field Name	Type	Reset	Description
[31:30]	address[33:32]	RO	0x0	Fixed to 0
[29:0]	address[31:2]	RW	Undefined	Determines whether an address matches the address [31:2].

In NS31A, functions are expanded as follows.

- Since NS31A uses a 32-bit address, the address [33:32] is fixed to 0. Write operation is ignored.
- The pmpaddr register (0-15) to be implemented depends on the number of entries specified in configuration parameter P_PMPNUM. If any pmpaddr register corresponding to an entry other than the specified ones is accessed, an illegal instruction exception occurs.

Write to pmpaddr0-15 is valid only if the corresponding entry is unlocked.
If an invalid write access is performed, no interrupt or exception will occur.

- For NAPOT, writing to pmpaddrn is valid if pmpncfg.L=0.
- For TOR, writing to pmpaddrn and pmpaddrn-1 is valid if pmpncfg.L=0.

3.7.2.8. Debug/Trace Registers

3.7.2.8.1. tselect

NS31A has two sets of trigger registers (tdata1, tdata2, and tinfo) that control the generation of event triggers for hart debugging. tselect sets an index for selecting a trigger register set.
It is a CSR defined in the RISC-V Debug Support Spec.
It is a register that can be read/written in the Machine mode and Debug mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	index
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 47. Debug/Trace trigger select register (tselect)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0x0	Reserved
[0]	index	RW	0x0	index of Trigger Registers Set 1: 2nd Trigger Registers Set 0: 1st Trigger Registers Set

3.7.2.8.2. tdata1

tdata1 is the 1st CSR for controlling the generation of event triggers for debugging the hart.
It is a CSR defined in the RISC-V Debug Support Spec.
It is a register that can be read/written in the Machine mode and Debug mode. However, when tdata1.dmode is 1, write in the Machine mode is ignored.
The type field of NS31A is fixed to 2. Even when a value other than 2 is written, the type cannot be changed.
Also, data match is not supported. When the type field is 2, it indicates that the address match trigger is set.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	type				dmode	maskmax						hit	select	timing	sizelo	
Reset:	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RW	RO	RO	RO	RO	RO	RO	RW	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	action				chain	match				m	—	—	u	execute	store	load
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RW	RW	RW	RW

Table 48. Debug match control register (tdata1_mcontrol) (type field: 2)

Bits	Field Name	Type	Reset	Description
[31:28]	type	RO	0x2	Trigger Type of tselect 2 : Address/Data Match Trigger (NS31A does not support Data match.)
[27]	dmode	RW	0x0	Debug only access mode 1: Write to tdata1 and tdata2 is possible only when the hart is in Debug mode. Write is ignored when the hart is not in Debug mode. 0: Write to tdata1 and tdata2 is possible even when the hart is not in Debug mode. Write to this bit is possible only when the hart is in Debug mode, regardless of the value of dmode.
[26:21]	maskmax	RO	0x0	Largest supported NAPOT range Specifies the maximum range of NAPOT supported by NS31A when the match field is 1. It specifies the value of log2 for the number of bytes within the range. In NS31A, this field is fixed to 0, which means that match 1 (NAPOT) is not supported.
[20]	hit	RW	0x0	Trigger Hit. When this trigger matches, this bit is set.
[19]	select	RO	0x0	0 : match on the address only.
[18]	timing	RO	0x0	Timing of breakpoint 0 : An action for a trigger is performed immediately before the instruction that caused the trigger is executed. (Note that it is performed after all of the preceding instructions have been committed.)
[17:16]	sizelo	RO	0x0	match size 0 : any size memory access The access size is not relevant.
[15:12]	action	RW	0x0	Trigger action 2-15 : Reserved 1 : The mode shifts to the DB-Halt mode. (When dmode of the trigger register is 1) 0 : Throw a breakpoint exception. (Purpose: When an external debugger is not used) If a Reserved value is written, 1 or 0 is set. Writing 0 to dmode causes this field to be cleared to 0.
[11]	chain	RW	0x0	Chain condition 1 : While this trigger does not match, a match is ignored even if the trigger matches in the next index (tselect). 0 : If this trigger matches, the set action is executed.
[10:7]	match	RW	0x0	Trigger match type 4-15: Matches never 3 : Matches when (value < tdata2 , unsigned) 2 : Matches when (value >= tdata2 , unsigned) 1: not supported "Power-of-2(NAPOT) range", Matches when values match exactly (value == tdata2) 0 : Matches when (value == tdata2)
[6]	m	RW	0x0	match on Machine-mode 1 : Enable 0 : Disable
[5:4]	-	RO	0x0	Reserved

Bits	Field Name	Type	Reset	Description
[3]	u	RW	0x0	match on User mode 1: Enable 0: Disable
[2]	execute	RW	0x0	address (or opcode) match on Instruction Fetch 1 : Enable 0 : Disable
[1]	store	RW	0x0	address (or data) match on Store/SC/AMO(Write) 1 : Enable 0 : Disable
[0]	load	RW	0x0	address (or data) match on Load/LR/AMO(Read) 1 : Enable 0 : Disable

3.7.2.8.3. tdata2

tdata2 is the 2nd CSR for controlling the generation of event triggers for debugging.
It is a CSR defined in the RISC-V Debug Support Spec.
It is a register that can be read/written in the Machine mode and Debug mode. However, when tdata1.dmode is 1, write in the Machine mode is ignored.
Since the type field of tdata1 in NS31A is fixed to 2, this register is used for setting the address that becomes the trigger generation condition of the address match trigger.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	data															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	data															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 49. Debug/Trigger 2nd trigger data register (tdata2)

Bits	Field Name	Type	Reset	Description
[31:0]	data	RW	0x0	trigger data for match

3.7.2.8.4. tinfo

tinfo is a register for indicating the type that is supported by the trigger register set selected in tselect.
It is a CSR defined in the RISC-V Debug Support Spec.
If type N is indicated, bit N is set to "1". If the trigger register set selected in tselect does not exist, bit 0 is set to "1".
NS31A supports only the address match trigger (type 2).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	info															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 50. Debug trigger info register (tinfo)

Bits	Field Name	Type	Reset	Description
[31:16]	-	RO	0x0	Reserved
[15:0]	info	RO	0x4	Trigger type supported by the trigger selected in tselect 4 : Address match trigger 1 : Trigger not set

3.7.2.9. Debug Mode Registers

3.7.2.9.1. dcsr

dcsr is a CSR for acquiring and controlling the status of the hart in the Debug mode.
It is a CSR defined in the RISC-V Debug Support Spec.
It is a register that can be read/written only in the Debug mode. Even if access is made in the Machine mode, an illegal

instruction exception occurs.
NS31A does not support the functions related to H-mode and S-mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	xdebugver				—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	ebreakm	—	—	ebreaku	stepie	stopcount	stoptime	cause			—	mprven	nmip	step	prv	
Reset:	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1
Type:	RW	RO	RO	RW	RO	RW	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW

Table 51. Debug control status register (dcsr)

Bits	Field Name	Type	Reset	Description
[31:28]	xdebugver	RO	0x4	Debug support 4 : Debug support exists as it is described in RISC-V Debug Support Spec.
[27:16]	-	RO	0x0	Reserved
[15]	ebreakm	RW	0x0	ebreak instruction in Machine-mode 1 : If the ebreak instruction is executed in the Machine mode, the mode shifts to the DB-Halt mode. 0 : If the ebreak instruction is executed in the Machine mode, the behavior described in RISC-V Priv. Arch. occurs.
[14:13]	-	RO	0x0	Reserved
[12]	ebreaku	RW	0x0	ebreak instruction in User mode 1: If the ebreak instruction is executed in the User mode, the mode shifts to the DB-Halt mode. 0: If the ebreak instruction is executed in the User mode, the behavior described in RISC-V Priv. Arch. occurs.
[11]	stepie	RO	0x0	disable interrupts during single stepping 0 : Interrupts that occur during single stepping are masked.
[10]	stopcount	RW	0x1	stop increment counters while in Debug-mode 1 : Stop incrementing minstret, mcycle and mhpmcounter3-10 while in the Debug mode. 0 : Do not stop incrementing minstret, mcycle and mhpmcounter3-10 while in the Debug mode.
[9]	stoptime	RO	0x0	This bit is not implemented.
[8:6]	cause	RO	0x0	Explains why Debug-mode was entered. 4 : Hart single stepped because dcsr's step-bit was set. (priority=0,lowest) 3 : Debugger requested entry to Debug-mode using haltreq. Or, Hart halted because it's part of halt group. (priority=1) 2 : Trigger caused a breakpoint exception. (priority=3,highest) 1 : ebreak instruction (priority=2) Other than the above : Reserved
[5]	-	RO	0x0	Reserved
[4]	mprven	RO	0x0	This bit is not implemented.
[3]	nmip	RO	0x0	NMI pending 1 : There is an NMI pending. 0 : There is no NMI pending.
[2]	step	RW	0x0	single step control If this field is set to 1 while in a non-Debug mode, the mode shifts to the DB-Halt mode immediately after an instruction is executed.

Bits	Field Name	Type	Reset	Description
[1:0]	prv	RW	0x3	Indicates the Privilege level when the mode shifts to the Debug mode 3 : Machine mode 0 : User mode If a value other than the above is written, the write is ignored. Upon return from Debug mode, execution starts in the mode that is set with these bits.

3.7.2.9.2. dpc

dpc is a CSR for holding the PC when the mode is shifted to the Debug mode.
It is set to the PC value of the hart when returned from the Debug mode.
It is a CSR defined in the RISC-V Debug Support Spec.
It is a register that can be read/written only in the Debug mode. Even if access is made in the Machine mode, an illegal instruction exception occurs.
For details of cfg_resetvec, see Section 11.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	dpc															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	dpc															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

(*1) cfg_resetvec

Table 52. Debug PC register (dpc)

Bits	Field Name	Type	Reset	Description
[31:0]	dpc	RW	cfg_resetvec	address of the next instruction to be executed. When the cause of dcsr is ebreak, the address of the ebreak instruction is stored. When the cause of dcsr is a single step, the address of the next instruction to be executed after the instruction that is executed with a single step is stored. When the cause of dcsr is a trigger, the address of the instruction executed when the trigger occurred is stored. When the cause of dcsr is haltreq, the address of the next instruction to be executed after the instruction executed when the shift to DB-Halt occurred is stored.

3.7.2.9.3. dscratch0

dscratch0 is a CSR for scratch used in the Debug mode.
It is a CSR defined in the RISC-V Debug Support Spec.
It is a register that can be read/written only in the Debug mode. Even if access is made in the Machine mode, an illegal instruction exception occurs.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	dscratch0															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	dscratch0															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x: Undefined

Table 53. Debug scratch register 0 (dscratch0)

Bits	Field Name	Type	Reset	Description
[31:0]	dscratch0	RW	Undefined	scratch register for Debug-mode.

3.7.2.10. Machine Counter/Timers

3.7.2.10.1. mcycle

The machine cycle counter (mcycle) is a 64-bit counter for counting the number of execution cycles.
It holds the lower 32 bits of the count value. To hold the upper 32 bits, mcycleh is used together.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The same value as this register can be read from cycle.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mcycle															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mcycle															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 54. Machine cycle counter (mcycle)

Bits	Field Name	Type	Reset	Description
[31:0]	mcycle	RW	0x0	lower 32bits of a count of the number of clock cycles that the core has executed.

Table 55 shows the relationship between mcycle(h) and other CSRs.

Table 55. mcycle(h) with other CSRs

CSR	Function
mcycle	lower 32bits counter for mcycle
mcylceh	upper 32bits counter for mcycleh
mcountinhibit.CY	Count control of mcycle(h) for Non-Debug Mode
dcsr.stopcount	Count control of minstrest(h)/mcycle(h) for Debug Mode

Table 56 shows the count conditions.
For the notes on the Halt and Resume operations, see Section 3.7.2.9.1.

Table 56. mcycle(h) behavior by setting other CSRs

Hart State	CSR.mcountinhibit.CY	CSR.dcsr.stopcount	Behavior of Shared Counter
Run	0	Don't care	Increment
DB-Run or DB-Halt	Don't care	0	Increment
Others			Not increment

3.7.2.10.2. minstret

The machine instruction-retired counter (minstret) is a 64-bit counter for counting the number of instructions that have been executed in the hart.
It holds the lower 32 bits of the count value. To hold the upper 32 bits, minstreth is used together.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The same value as this register can be read from instret.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	minstret															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	minstret															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 57. Machine instruction-retired counter (minstret)

Bits	Field Name	Type	Reset	Description
[31:0]	minstret	RW	0x0	lower 32bits of a count of the number of instructions that hart has retired.

Table 58 shows the relationship between minstret(h) and other CSRs.

Table 58. minstret(h) with other CSRs

CSR	Function
minstret	lower 32bits counter for minstret
minstreth	upper 32bits counter for minstreth
mcountinhibit.IR	Count control of minstret(h) for Non-Debug Mode
dcsr.stopcount	Count control of minstrest(h)/myclce(h) for Debug Mode

Table 59 shows the count conditions.
For the notes on the Halt and Resume operations, see Section 3.7.2.9.1.

Table 59. minstret(h) behavior by setting other CSRs

Hart State	CSR.mcountinhibit.IR	CSR.dcsr.stopcount	Behavior of Each Counters
Run	0	Don't care	Increment
DB-Run or DB-Halt	Don't care	0	Increment
Others			Not increment

3.7.2.10.3. mhpmcounter3-10

Machine Hardware Performance Monitor counters (mhpmcounter3 to mhpmcounter10) are 40-bit counters for counting the execution performance of the HART that has been set in mhpmevent3 to mhpmevent10.
mhpmcounter3 to mhpmcounter10 hold the lower 32 bits of the count value. To hold the upper 8 bits, mhpmcounter3h to mhpmcounter10h are used together.
For details, see Section 3.7.3.
It is a register that can be read/written in the Machine mode. The same value as this register can be read from hpmcounter3 to hpmcounter10. The number of registers that will be actually implemented can be set by setting configuration parameter P_HPMNUM.
Accessing an unimplemented register causes an Illegal Instruction Exception to be thrown.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mhpmcounter															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mhpmcounter															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 60. Machine performance-monitoring counter (mhpmcounter3-10)

Bits	Field Name	Type	Reset	Description
[31:0]	mhpmcounter	RW	0x0	lower 32bits of Hardware Performance Monitor.

Table 61 shows the count conditions.
For the notes on the Halt and Resume operations, see Section 3.7.2.9.1.

Table 61. mhpmcounter(h) behavior by setting other CSRs

Hart State	CSR.mcountinhibit.HPM3-10	CSR.dcsr.stopcount	Behavior of Each Counters
Run	0	Don't care	Increment
DB-Run or DB-Halt	Don't care	0	Increment
Others			Not increment

3.7.2.10.4. mcycleh

The machine cycle counter (mcycleh) is a 64-bit counter for counting the number of execution cycles.
It holds the upper 32 bits of the count value.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The same value as this register can be read from cycleh.

For the conditions for starting and ending the count, see Section 3.7.2.10.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mcycleh															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mcycleh															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 62. Upper 32 bits of mcycle (mcycleh)

Bits	Field Name	Type	Reset	Description
[31:0]	mcycleh	RW	0x0	upper 32bits of a count of the number of clock cycles that the core has executed.

3.7.2.10.5. minstreth

The machine instruction-retired counter (minstreth) is a 64-bit counter for counting the number of instructions that have been executed in the hart.
It holds the upper 32 bits of the count value.
It is a CSR defined in RISC-V Priv. Arch.
It is a register that can be read/written in the Machine mode. The same value as this register can be read from instreth.

For the conditions for starting and ending the count, see Section 3.7.2.10.2.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	minstreth															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	minstreth															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 63. Upper 32 bits of minstret (minstreth)

Bits	Field Name	Type	Reset	Description
[31:0]	minstreth	RW	0x0	upper 32bits of a count of the number of instructions that hart has retired.

3.7.2.10.6. mhpmcounter3h-10h

The Machine Hardware Performance Monitor counters (mhpmcounter3h to mhpmcounter10h) are 40-bit counters for counting the execution performance of the HART that has been set in mhpmevent3 to mhpmevent10.
mhpmcounter3h to mhpmcounter10h hold upper 8 bits of the count value.
For details, see Section 3.7.3.
It is a register that can be read/written in the Machine mode. The same value as this register can be read from hpmcounter3h to hpmcounter10h. The number of registers that will be actually implemented can be set by setting configuration parameter P_HPMNUM.
Accessing an unimplemented register causes an Illegal Instruction Exception to be thrown.

For the conditions for starting and stopping the count, see Section 3.7.2.10.3.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	mhpmcounterh							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW

Table 64. Upper 8 bits of mhpmcountern (mhpmcounterh3h-10h)

Bits	Field Name	Type	Reset	Description
[31:8]	-	RO	0x0	Reserved
[7:0]	mhpmcounterh	RW	0x0	upper 8bits of Hardware Performance Monitor.

3.7.2.11. User Counter/Timers

3.7.2.11.1. cycle

cycle is a CSR for acquiring the number of execution cycles by using the RDCYCLE instruction.
Like mcycle, this register allows the lower 32 bits of the count value to be read. If counting is performed in 64-bit width, not 32-bit width, cycleh is used together.
It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	cycle															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	cycle															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 65. Cycle Counter for RDCYCLE instruction (cycle)

Bits	Field Name	Type	Reset	Description
[31:0]	cycle	RO	0x0	lower 32 bits of a count of the number of clock cycles that the core has executed.

3.7.2.11.2. time

time is a CSR for acquiring the time by using the RDTIME instruction.
It allows the lower 32 bits of the timer counter in the ACLINT to be read. To acquire the time value of 64-bit width, timeh is used together.
It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	time															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	time															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 66. Machine cycle counter (time)

Bits	Field Name	Type	Reset	Description
[31:0]	time	RO	0x0	lower 32bits of a real-time counter.

3.7.2.11.3. instret

instret is a CSR for acquiring the number of instructions that have been executed in a hart by using the RDINSTRET instruction.
Like minstret, this register allows the lower 32 bits of the count value to be read. If counting is performed in 64-bit width, not 32-bit width, instreth is used together.

It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	instret															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	instret															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 67. Instructions-retired counter for RDINSTRET instruction (instret)

Bits	Field Name	Type	Reset	Description
[31:0]	instret	RO	0x0	lower 32bits of a count of the number of instructions that hart has retired.

3.7.2.11.4. hpmcounter3-10

Each of hpmcounter3 to hpmcounter10 is a CSR for acquiring the Hardware Performance Counter value executed by the HART.
Like mhpmcounter3 to mhpmcounter10, this register allows the lower 32 bits of the count value to be read.
It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User Mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.
Accessing an unimplemented register causes an Illegal Instruction Exception to be thrown.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	hpmcounter															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	hpmcounter															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 68. Performance-monitoring counter (hpmcounter3-10)

Bits	Field Name	Type	Reset	Description
[31:0]	hpmcounter	RO	0x0	lower 32bits of Hardware Performance Monitor.

3.7.2.11.5. cycleh

cycleh is a CSR for acquiring the number of execution cycles by using the RDCYCLEH instruction.
Like mcycleh, this register allows the upper 32 bits of the count value to be read.
It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	cycleh															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	cycleh															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 69. Upper 32 bits of cycle (cycleh)

Bits	Field Name	Type	Reset	Description
[31:0]	cycleh	RO	0x0	upper 32bits of a count of the number of clock cycles that the core has executed.

3.7.2.11.6. timeh

timeh is a CSR for acquiring the time by using the RDTIMEH instruction.
It allows the upper 32 bits of the timer counter in the ACLINT to be read.
It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	timeh															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	timeh															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 70. Upper 32 bits of time (timeh)

Bits	Field Name	Type	Reset	Description
[31:0]	timeh	RO	0x0	upper 32bits of a real-time counter.

3.7.2.11.7. instreth

instreth is a CSR for acquiring the number of instructions that have been executed in the hart by using the RDINSTRETH instruction. Like minstreth, this register allows the upper 32 bits of the count value to be read.
It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	instreth															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	instreth															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 71. Upper 32 bits of instret (instreth)

Bits	Field Name	Type	Reset	Description
[31:0]	instreth	RO	0x0	upper 32bits of a count of the number of instructions that hart has retired.

3.7.2.11.8. hpmcounter3h-10h

Each of hpmcounter3h to hpmcounter10h is a CSR for acquiring the Hardware Performance Counter value executed by the hart.
Like mhpmcounter3h to mhpmcounter10h, this register allows upper 8 bits of the count value to be read.
It is a CSR defined in RISC-V Priv. Arch.
This register can be accessed with the CSR instructions in User Mode by making the appropriate setting in mcounteren.
For details, see Section 3.7.2.2.5.
Accessing an unimplemented register causes an Illegal Instruction Exception to be thrown.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	hpmcounterh							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW

Table 72. Upper 8 bits of hpmcountern (hpmcounter3h-10h)

Bits	Field Name	Type	Reset	Description
[31:8]	-	RO	0x0	Reserved
[7:0]	hpmcounterh	RO	0x0	upper 8bits of Hardware Performance Monitor.

3.7.2.12. Machine Information Registers

3.7.2.12.1. mvendorid

mvendorid is a CSR for acquiring the JEDEC manufacturer ID of the provider of the core.
In NS31A, it is fixed to the JEDEC manufacturer ID of NSITEXE.
It is a CSR defined in RISC-V Priv. Arch.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	Bank															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	Bank									Offset						
Reset:	0	0	0	0	0	1	0	0	1	1	1	0	1	1	1	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 73. Machine Vendor ID Register (mvendorid)

Bits	Field Name	Type	Reset	Description
[31:7]	Bank	RO	0x9	the number of one-byte continuation codes(0x7f)
[6:0]	Offset	RO	0x6F	final byte is used to encode

3.7.2.12.2. marchid

marchid is a CSR for acquiring the ID indicating the basic micro-architecture of the core.
It is fixed to Architecture ID defined in the specifications for NS31A.
It is a CSR defined in RISC-V Priv. Arch.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	archid															
Reset:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	archid															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 74. Machine Architecture ID (marchid)

Bits	Field Name	Type	Reset	Description
[31:0]	archid	RO	32’h80000005	Architecture ID: NS31A

3.7.2.12.3. mimpid

mimpid is a CSR for acquiring the ID indicating the version of processor implementation.
It is fixed to Implementation ID defined in the specifications for NS31A.
It is a CSR defined in RISC-V Priv. Arch.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	impid															
Reset:	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	impid															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 75. Machine Implementation ID Register (mimpid)

Bits	Field Name	Type	Reset	Description
[31:0]	impid	RO	32’h02000004	Implementation ID

3.7.2.12.4. mhartid

mhartid is a CSR for acquiring the ID of the hart.
It is a CSR defined in RISC-V Priv. Arch.
In NS31A, 0 to 31 can be assigned as hart IDs.
The values are fixed to the IDs specified in configuration parameter P_HARTID.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	Hart ID															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	Hart ID															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) depend on P_HARTID

Table 76. Hart ID Register (mhartid)

Bits	Field Name	Type	Reset	Description
[31:0]	Hart ID	RO	P_HARTID setting value	Hart ID

3.7.2.12.5. mconfigptr

mconfigptr is a pointer to configuration data structure which is used by Software.
It is a CSR defined in RISC-V Priv. Arch.
It is fixed to 0 since configuration data structure is not implemented in NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	configptr															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	configptr															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 77. Machine Configuration Pointer Register (mconfigptr)

Bits	Field Name	Type	Reset	Description
[31:0]	configptr	RO	0	Configuration Pointer Register

3.7.3. Hardware Performance Monitor (HPM)

NS31A offers additional counters, mhpmcounter3 to mhpmcounter10, as Hardware Performance Monitor (HPM), in addition to mcycle and minstret. The number of additional counters that will be actually implemented can be set by parameter P_HPMNUM.
The specific event categories to be acquired by HPM may be defined in the future as part of the RISC-V specifications, but for NS31A, the specifications are implementation-dependent.

3.7.3.1. Implementation-dependent part specifications

Event Selector Registers

They are registers of 32-bit width, mhpmevent3 to mhpmevent10, and are used to specify the acquired event of the related mhpmcounter3(h) to mhpmcounter10(h). Each of the registers is divided into two fields. The lower 8 bits represent the class field for specifying an event class, and the upper 24 bits represent the mask field for each event. For details, see Section 3.7.2.4.2.

Event Selector Encodeing

An event category can be specified with mhpmeventn[7:0] (Event Class). In the Bits column of the encoding table shown below, the contents of mhpmeventn[31:8] (Event Mask) in the event to be acquired can be defined.

NS31A supports Event Classes between 0x0 to 0x2, but does not support some events.

Table 78. HPM Encoding (mhpmeventn[7:0] = 0x0)

Bits	Description
8	Exception taken
9	Integer load instruction retired
10	Integer store instruction retired
11	Atomic memory operation retired
12	System instruction retired
13	Integer arithmetic instruction retired
14	Conditional branch retired
15	JAL instruction retired
16	JALR instruction retired
17	Integer multiplication instruction retired
18	Integer division instruction retired
19	Floating-point load instruction retired
20	Floating-point store instruction retired
21	Floating-point addition retired
22	Floating-point multiplication retired
23	Floating-point fused multiply-add retired
24	Floating-point division or square-root retired
25	Other floating-point instruction retired

Table 79. HPM Encoding (mhpmeventn[7:0] = 0x1)

Bits	Description
8	not supported (Address-generation interlock)
9	not supported (Long-latency interlock)
10	not supported (CSR read interlock)
11	not supported (Instruction cache/ILM busy)
12	not supported (DLM busy)
13	Branch direction misprediction
14	Branch/jump target misprediction
15	not supported (Pipeline flush from CSR write)
16	Pipeline flush from other event
17	Integer multiplication interlock
18	not supported (Floating-point interlock)

Table 80. HPM Encoding (mhpmeventn[7:0] = 0x2)

Bits	Description
8	Instruction cache miss
9	not supported (Data cache miss or memory-mapped I/O access)
10	not supported (Data cache write-back)
11	not supported (Instruction TLB miss)
12	not supported (Data TLB miss)

3.8. Memory Mapped Control Registers

This section describes the memory-mapped registers included in the processor system.
The following registers are included.

- Memory Mapped Control Register
- Hart Context Register

The Memory Mapped Control register is a control register for accessing exclusive monitor, link bit, instruction cache, and BTB information.
The Hart context register is a control register for controlling the context of the hart as NS31A-specific extension specifications. Both of them are NS31A-specific custom-defined registers.

3.8.1. Memory Mapped Control Register

This section describes the Memory Mapped Control registers.

3.8.1.1. Memory Mapped Control register

Table 81 shows the list of the Memory Mapped Control registers.
The address of each register is an offset value from the Base Address (P_ADDR_BASE_REG).

Table 81. Memory Mapped Control Register List

Category	Register Name	Symbol	Address	Access	Access Protection
EXMON [*1]	Exclusive Monitor Entry0's Valid bit Register	EXCLMONVALIDBITENTRY0	+0000H	32	-
	Exclusive Monitor Entry0's Address Register	EXCLMONADDRESSENTRY0	+0004H	32	-
	Exclusive Monitor Entry0's ID Register	EXCLMONIDENTRY0	+0008H	32	-
	Exclusive Monitor Entry0's Sideband Register	EXCLMONSIDEBANDENTRY0	+000CH	32	-
	Exclusive Monitor Entry1's Valid bit Register	EXCLMONVALIDBITENTRY1	+0010H	32	-
	Exclusive Monitor Entry1's Address Register	EXCLMONADDRESSENTRY1	+0014H	32	-
	Exclusive Monitor Entry1's ID Register	EXCLMONIDENTRY1	+0018H	32	-
	Exclusive Monitor Entry1's Sideband Register	EXCLMONSIDEBANDENTRY1	+001CH	32	-
LL	1st Hart's Link Bit Register	LINKBIT0	+0100H	32	-
	1st Hart's Link Address Register	LINKADDR0	+0104H	32	-
ICRAM [*1]	Inst. Cache RAM access control Register	L1CRACTRL	+0600H	32	-
	Inst. Cache RAM access target Register	L1CRATGT	+0604H	32	-
	Inst. Cache RAM access data0 Register	L1CRADAT0	+0608H	32	-
	Inst. Cache RAM access data1 Register	L1CRADAT1	+060CH	32	-
BTB [*1]	BTB Access Control Register	BTBACTRL	+0800H	32	-
	BTB Access Target Register	BTBATGT	+0804H	32	-
	BTB Access Data Register Register	BTBADAT	+0808H	32	-

*1 If the register is accessed when this function is not supported for the value set in the configuration parameter, a bus error response (DECERR) occurs.

EXMON is not supported in the AHB configuration. It is necessary to provide the corresponding function at the end of the external bus.

Access under any of the following conditions causes a bus error response (SLVERR).

- Write access with a size smaller than 32 bits
- Access in U-mode from the NS31A hart (including when MPRV is 1 and MPP is 0 in M-mode)
- Access from the Front Port Bus Slave, with AxPROT[0] set to 0

Access under the following condition causes a bus error response (DECERR).

- Access to an address that is not supported

In the case of AHB configuration, replace "SLVERR" and "DECERR" with "bus error".

3.8.2. Hart Context Register

This section describes the Hart Context registers.

Access under any of the following conditions causes a bus error response (SLVERR).

- Write access with a size smaller than 32 bits
- Access in U-mode from the NS31A hart (including when MPRV is 1 and MPP is 0 in M-mode)
- Access from the Front Port Bus Slave, with AxPROT[0] set to 0 (Unprivileged)
- Access to an address that is not supported
- Access to addresses not aligned with 4 bytes

In the case of AHB configuration, replace "SLVERR" and "DECERR" with "bus error".

3.8.2.1. Hart context register list

The list of the Hart context registers is shown below.
The address of each register is an offset value from the Base Address (P_ADDR_BASE_HCR).

Table 82. Hart Context Register List

Category	Register Name	Symbol	Address	Access	Access Protection
Exception Error Address of first imprecise data access fault	Captured Error Address Register	MEECADDR	+1F0CH	32	-
	Unlock control for meecaddr Register	MEECUNLOCK	+1F10H	32	-
Hart control	Machine Hart PC Register	MHTPC	+2F00H	32	-
	Hart Enable Control Status Register	MHTENABLE	+2F40H	32	-
	Hart Run Control Status Register	MHTSTATUS	+2F80H	32	-
L1 Instruction Cache control	Level-1 Inst. Cache enable control Register	ML1CACHESYSCTRL (*1)	+2FA0H	32	-
	Level-1 Inst. Cache operation Register	ML1CACHEOPERATION (*1)	+2FA4H	32	-
	Level-1 Inst. Prefetch Control Register	ML1CACHEPREFCFG (*1)	+2FA8H	32	-
	Branch Prediction System Control Register	MBPSYSCTRL (*1)	+2FB8H	32	-
	Branch Prediction Operation Register	MBPOPERATION (*1)	+2FBCH	32	-
Hart Fetch address Register	Hart Fetch address Register	MHTFETCHADDR	+3F00H	32	-

*1 This function may not be supported depending on the configuration parameter. In this case, access to this register causes a bus error response.

3.8.3. Register descriptions

This section describes the specifications of the individual memory-mapped registers included in the processor system.

3.8.3.1. Memory Mapped Control Registers

3.8.3.1.1. EXCLMONVALIDBITENTRYn

EXCLMONVALIDBITENTRYn are registers for accessing Valid-Bits of entries in the DLM exclusive monitor for self-diagnosis.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	Valid
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 83. Exclusive Monitor Entry N’s Valid-Bit (EXCLMONVALIDBITENTRYn)(n=0-1)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0x0	Reserved
[0]	Valid	RW	0x0	Valid-Bit of ENTRYn

3.8.3.1.2. EXCLMONADDRESSENTRYn

EXCLMONADDRESSENTRYn are registers for accessing the address information of entries in the DLM exclusive monitor for self-diagnosis.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	address															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	address															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x :Undefined

Table 84. Exclusive Monitor Entry N’s Address (EXCLMONADDRESSENTRYn)

Bits	Field Name	Type	Reset	Description
[31:0]	address	RW	Undefined	Address of an ENTRYn bus command

3.8.3.1.3. EXCLMONIDENTRYn

EXCLMONIDENTRYn are registers for accessing the tag information of entries in the DLM exclusive monitor for self-diagnosis. (n = 0 - 1)

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	ID[19:16]			
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	ID[15:0]															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x: Undefined

Table 85. Exclusive Monitor Entry N’s ID (EXCLMONIDENTRYn)

Bits	Field Name	Type	Reset	Description
[31:20]	-	RO	0x0	Reserved
[19:0]	ID	RW	Undefined	Request ID of an ENTRYn bus command

3.8.3.1.4. EXCLMONSIDEBANDENTRYn

EXCLMONSIDEBANDENTRYn are registers for accessing the sideband information (access size) of entries in the DLM exclusive monitor for self-diagnosis. (n = 0 - 1)

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	ID	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW

x: undefined

Table 86. Exclusive Monitor Entry N’s Sideband (EXCLMONSIDEBANDENTRYn)

Bits	Field Name	Type	Reset	Description
[31:2]	-	RO	0x0	Reserved
[1:0]	ID	RW	Undefined	Sideband information (access size) of the request command of an ENTRYn bus command 00: 1-Byte 01: 2-Byte 10: 4-Byte 11: Reserved

3.8.3.1.5. LINKBIT0

LINKBIT0 is a register for accessing the link bit for self-diagnosis.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Link Bit
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

x :Undefined

Table 87. Link Bit (LINKBIT0)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0x0	Reserved
[0]	Link Bit	RW	Undefined	Link bit

3.8.3.1.6. LINKADDR0

LINKADDR0 is a register for accessing the link address for self-diagnosis.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	Link Address															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	Link Address															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x：Undefined

Table 88. Link Address (LINKADDR0)

Bits	Field Name	Type	Reset	Description
[31:0]	Link Address	RW	Undefined	Link address

3.8.3.1.7. L1CRACTRL

L1CRACTRL is a register for accessing the internal RAM of the Instruction Cache for self-diagnosis.
It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC ≥ 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	RAC	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO

Table 89. Access for Diagnosis of Instruction Cache (internal RAM) (L1CRACTRL)

Bits	Field Name	Type	Reset	Description
[31:2]	-	RO	0	Reserved
[1:0]	RAC	WO	0	RAM Access Control 0: Disabled 1: Execute Read access 2 or 3: Execute Write access

3.8.3.1.8. L1CRATGT

L1CRATGT is a register for setting the access target RAM and address of the Instruction Cache for self-diagnosis.
It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC ≥ 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	RAMSEL		
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	INDEX										
Reset:	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
Type:	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x: undefined

Table 90. Access for Diagnosis of Instruction Cache (Target RAM/address) (L1CRATGT)

Bits	Field Name	Type	Reset	Description
[31:19]	-	RO	0	Reserved
[18:16]	RAMSEL	RW	Undefined	Select the access target RAM 0:TagRAM way0 1:TagRAM way1 2:DataRAM Bank0 3:DataRAM Bank1 4:Line Valid-bit 5:LRU-bit 6:No operation 7:No operation
[15:11]	-	RO	0	Reserved

Bits	Field Name	Type	Reset	Description
[10:0]	INDEX	RW	Undefined	Specify the access target address If an invalid address is specified, 0 is read for a read access and a write access is ignored. The address bits valid for each access target RAM are as follows: **DataRAM:** INDEX[P_CACHE_SIZE_IC+6:0] **TagRAM:** INDEX[P_CACHE_SIZE_IC+4:0] **Line Valid-bit:** INDEX[P_CACHE_SIZE_IC-2:0] only when P_CACHE_SIZE_IC is equal to or greater than 2(*1) **LRU-bit:** INDEX[P_CACHE_SIZE_IC-2:0] only when P_CACHE_SIZE_IC is equal to or greater than 2(*1)

(*1) Specify 0x0 when P_CACHE_SIZE_IC = 1.

3.8.3.1.9. L1CRADAT0

L1CRADAT0 is a register for storing the lower 32 bits of the internal RAM access data of the Instruction Cache for self-diagnosis.
It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC ≥ 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	DATA															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	DATA															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x:Undefined

Table 91. Access for Diagnosis of Instruction Cache (RAM access data, lower 32-bit)(L1CRADAT0)

Bits	Field Name	Type	Reset	Description
[31:0]	DATA	RW	Undefined	RAM access data (SeeTable 92)

Table 92. Data stored in L1CRADAT0

Condition		Data stored	
For DataRAM read		[31:0]	Lower 32 bits of Read Data
For DataRAM write		[31:0]	Lower 32 bits of Write Data
For TagRAM read	P_CACHE_SIZE_IC = 1	[31:22]	Reserved (0)
		[21]	Read Data of Valid
		[20:1]	Read Data of Tag
		[0]	Always 0
	P_CACHE_SIZE_IC=n (n = 2, 3, 4)	[31:22]	Reserved (0)
		[21]	Read Data of Valid
		[20:(n-1)]	Read Data of Tag
		[(n-2):0]	Always 0

Condition		Data stored	
For TagRAM write	P_CACHE_SIZE_IC = 1	[31:22]	Reserved (disabled)
		[21]	Write Data of Valid
		[20:1]	Write Data of Tag
		[0]	Disabled
	P_CACHE_SIZE_IC=n (n = 2, 3, 4)	[31:22]	Reserved (disabled)
		[21]	Write Data of Valid
		[20:(n-1)]	Write Data of Tag
		[(n-2):0]	Disabled
For Line Valid-bit read	When L1CRATGT INDEX is 0x0	[31:0]	Read Data of the Line Valid-bit information of cache lines 0-31 [*1]
	When L1CRATGT INDEX is 0x1	[31:0]	Read Data of the Line Valid-bit information of cache lines 64-95 [*2]
	When L1CRATGT INDEX is 0x2	[31:0]	Read Data of the Line Valid-bit information of cache lines 128-159 [*3]
	When L1CRATGT INDEX is 0x3	[31:0]	Read Data of the Line Valid-bit information of cache lines 192-223 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Read Data of the Line Valid-bit information of cache lines 256-287 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Read Data of the Line Valid-bit information of cache lines 320-351 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Read Data of the Line Valid-bit information of cache lines 384-415 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Read Data of the Line Valid-bit information of cache lines 448-479 [*4]
For Line Valid-bit write	When L1CRATGT INDEX is 0x0	[31:0]	Write Data of the Line Valid-bit information of cache lines 0-31 [*1]
	When L1CRATGT INDEX is 0x1	[31:0]	Write Data of the Line Valid-bit information of cache lines 64-95 [*2]
	When L1CRATGT INDEX is 0x2	[31:0]	Write Data of the Line Valid-bit information of cache lines 128-159 [*3]
	When L1CRATGT INDEX is 0x3	[31:0]	Write Data of the Line Valid-bit information of cache lines 192-223 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Write Data of the Line Valid-bit information of cache lines 256-287 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Write Data of the Line Valid-bit information of cache lines 320-351 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Write Data of the Line Valid-bit information of cache lines 384-415 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Write Data of the Line Valid-bit information of cache lines 448-479 [*4]
For LRU-bit read	When L1CRATGT INDEX is 0x0	[31:0]	Read Data of the LRU-bit information of cache lines 0-31 [*1]
	When L1CRATGT INDEX is 0x1	[31:0]	Read Data of the LRU-bit information of cache lines 64-95 [*2]
	When L1CRATGT INDEX is 0x2	[31:0]	Read Data of the LRU-bit information of cache lines 128-159 [*3]
	When L1CRATGT INDEX is 0x3	[31:0]	Read Data of the LRU-bit information of cache lines 192-223 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Read Data of the LRU-bit information of cache lines 256-287 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Read Data of the LRU-bit information of cache lines 320-351 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Read Data of the LRU-bit information of cache lines 384-415 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Read Data of the LRU-bit information of cache lines 448-479 [*4]
For LRU-bit write	When L1CRATGT INDEX is 0x0	[31:0]	Write Data of the LRU-bit information of cache lines 0-31 [*1]
	When L1CRATGT INDEX is 0x1	[31:0]	Write Data of the LRU-bit information of cache lines 64-95 [*2]
	When L1CRATGT INDEX is 0x2	[31:0]	Write Data of the LRU-bit information of cache lines 128-159 [*3]
	When L1CRATGT INDEX is 0x3	[31:0]	Write Data of the LRU-bit information of cache lines 192-223 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Write Data of the LRU-bit information of cache lines 256-287 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Write Data of the LRU-bit information of cache lines 320-351 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Write Data of the LRU-bit information of cache lines 384-415 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Write Data of the LRU-bit information of cache lines 448-479 [*4]

P_CACHE_SIZE_IC is a configuration parameter (see Table 195).

*1 (Cache line number = Bit position) (Valid when P_CACHE_SIZE_IC = 1, 2, 3, 4)
*2 (Cache line number = 64 + Bit position) (Valid when P_CACHE_SIZE_IC = 2, 3, 4)
*3 (Cache line number = 128 + Bit position) (Valid when P_CACHE_SIZE_IC = 3, 4)
*4 (Cache line number = 256 + Bit position) (Valid when P_CACHE_SIZE_IC = 4)

3.8.3.1.10. L1CRADAT1

L1CRADAT1 is a register for storing the upper 32 bits of the internal RAM access data of the Instruction Cache for self-diagnosis.
It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC ≥ 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	DATA															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	DATA															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x:Undefined

Table 93. Access for Diagnosis of Instruction Cache (RAM access data, higher 32-bit)(L1CRADAT1)

Bits	Field Name	Type	Reset	Description
[31:0]	DATA	RW	Undefined	RAM access data (SeeTable 94)

Table 94. Data stored in L1CRADAT1

Condition		Data stored	
For DataRAM read		[31:0]	Upper 32 bits of Read Data
For DataRAM write		[31:0]	Upper 32 bits of Write Data
For TagRAM read		[31:0]	Reserved(0)
For TagRAM write		[31:0]	Reserved(disabled)
For Line Valid-bit read	when L1CRATGT INDEX is 0x0	[31:0]	Read Data of the Line Valid-bit information of cache lines 32-63 [*1]
	when L1CRATGT INDEX is 0x1	[31:0]	Read Data of the Line Valid-bit information of cache lines 96-127 [*2]
	when L1CRATGT INDEX is 0x2	[31:0]	Read Data of the Line Valid-bit information of cache lines 160-191 [*3]
	when L1CRATGT INDEX is 0x3	[31:0]	Read Data of the Line Valid-bit information of cache lines 224-255 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Read Data of the Line Valid-bit information of cache lines 288-319 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Read Data of the Line Valid-bit information of cache lines 352-383 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Read Data of the Line Valid-bit information of cache lines 416-447 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Read Data of the Line Valid-bit information of cache lines 480-511 [*4]

Condition		Data stored	
For Line Valid-bit write	when L1CRATGT INDEX is 0x0	[31:0]	Write Data of the Line Valid-bit information of cache lines 32-63 [*1]
	when L1CRATGT INDEX is 0x1	[31:0]	Write Data of the Line Valid-bit information of cache lines 96-127 [*2]
	when L1CRATGT INDEX is 0x2	[31:0]	Write Data of the Line Valid-bit information of cache lines 160-191 [*3]
	when L1CRATGT INDEX is 0x3	[31:0]	Write Data of the Line Valid-bit information of cache lines 224-255 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Write Data of the Line Valid-bit information of cache lines 288-319 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Write Data of the Line Valid-bit information of cache lines 352-383 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Write Data of the Line Valid-bit information of cache lines 416-447 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Write Data of the Line Valid-bit information of cache lines 480-511 [*4]
For LRU-bit read	when L1CRATGT INDEX is 0x0	[31:0]	Read Data of the LRU-bit information of cache lines 32-63 [*1]
	when L1CRATGT INDEX is 0x1	[31:0]	Read Data of the LRU-bit information of cache lines 96-127 [*2]
	when L1CRATGT INDEX is 0x2	[31:0]	Read Data of the LRU-bit information of cache lines 160-191 [*3]
	when L1CRATGT INDEX is 0x3	[31:0]	Read Data of the LRU-bit information of cache lines 224-255 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Read Data of the LRU-bit information of cache lines 288-319 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Read Data of the LRU-bit information of cache lines 352-383 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Read Data of the LRU-bit information of cache lines 416-447 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Read Data of the LRU-bit information of cache lines 480-511 [*4]
For LRU-bit write	when L1CRATGT INDEX is 0x0	[31:0]	Write Data of the LRU-bit information of cache lines 32-63 [*1]
	when L1CRATGT INDEX is 0x1	[31:0]	Write Data of the LRU-bit information of cache lines 96-127 [*2]
	when L1CRATGT INDEX is 0x2	[31:0]	Write Data of the LRU-bit information of cache lines 160-191 [*3]
	when L1CRATGT INDEX is 0x3	[31:0]	Write Data of the LRU-bit information of cache lines 224-255 [*3]
	When L1CRATGT INDEX is 0x4	[31:0]	Write Data of the LRU-bit information of cache lines 288-319 [*4]
	When L1CRATGT INDEX is 0x5	[31:0]	Write Data of the LRU-bit information of cache lines 352-383 [*4]
	When L1CRATGT INDEX is 0x6	[31:0]	Write Data of the LRU-bit information of cache lines 416-447 [*4]
	When L1CRATGT INDEX is 0x7	[31:0]	Write Data of the LRU-bit information of cache lines 480-511 [*4]

P_CACHE_SIZE_IC is a configuration parameter (See Table 195).

*1 (Cache line number = 32+ Bit position) (Valid when P_CACHE_SIZE_IC = 1, 2, 3, 4)
*2 (Cache line number = 96 + Bit position) (Valid when P_CACHE_SIZE_IC = 2, 3, 4)
*3 (Cache line number = 160 + Bit position) (Valid when P_CACHE_SIZE_IC = 3, 4)
*4 (Cache line number = 288 + Bit position) (Valid when P_CACHE_SIZE_IC = 4)

3.8.3.1.11. BTBACTRL

BTBACTRL is a register for accessing the Branch Target Buffer (BTB) in the branch prediction mechanism for self-diagnosis.
It is a register that can be read/written when configuration parameter P_BTB is 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	EAC	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO

Table 95. Access for Diagnosis of BTB (Access Control) (BTBACTRL)

Bits	Field Name	Type	Reset	Description
[31:2]	-	RO	0	Reserved
[1:0]	EAC	WO	0	Entry Access Control 0: Disabled 1: Executes Read access. 2, 3: Executes Write access.

3.8.3.1.12. BTBATGT

BTBATGT is a register for setting the access target and address of the BTB for self-diagnosis.
It is a register that can be read/written when configuration parameter P_BTB is 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	EntrySEL	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	INDEX				
Reset:	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW

x: Undefined

Table 96. Access for Diagnosis of BTB (Target Entry/index) (BTBATGT)

Bits	Field Name	Type	Reset	Description
[31:18]	-	RO	0	Reserved
[17:16]	EntrySEL	RW	Undefined	Selects the access target RAM. 0: TagAddress + Valid-bit 1: Data (TargetAddress) 2: Dynamic prediction bit and Hint information 3: No operation
[15:5]	-	RO	0	Reserved
[4:0]	INDEX	RW	Undefined	Access target address (BTB entry number)

3.8.3.1.13. BTBADAT

BTBADAT is a register for storing BTB data for self-diagnosis.
It is a register that can be read/written when configuration parameter P_BTB is 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	DATA															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	DATA															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

x: Undefined

Table 97. Access for Diagnosis of BTB (Entry access data)(BTBADAT)

Bits	Field Name	Type	Reset	Description
[31:0]	DATA	RW	Undefined	Entry access data * The access target in the Entry is TagAddress + Valid-bit. [31:1] : TagAddress [0] : Valid-bit * The access target in the Entry is Data (TargetAddress). [31:1] : TargetAddress[31:1] [0] : Fixed to 0 * The access target in the Entry is Hint information. [31:30]: Dynamic prediction bits [29]: Entry store instruction: (1: RV32C, 0: Others) [28:4] : Fixed to 0 [3:0] : Hint information

3.8.3.2. Hart Context Registers

3.8.3.2.1. meecaddr

meecaddr is a register for capturing an error address at which an imprecise data access fault occurs in a hart.
The value can be changed by software.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	Captured Error Address															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	Captured Error Address															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 98. Machine Exception Error Captured Address register (meecaddr)

Bits	Field Name	Type	Reset	Description
[31:0]	Captured Error Address	RW	0x0	Captured Error Address of first imprecise bus access fault. The address of the first imprecise bus error exception that occurs while the address capture is not locked is captured. After the reset is released, the address capture is not locked. Once the address is captured, the address capture is locked. In the locked state, data cannot be written. To unlock the capture, use System Reset or write to the meecunlock register.

3.8.3.2.2. meecunlock

meecunlock is a register for releasing the locked meecaddr in a hart.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	unlock															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	unlock															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Table 99. Machine Exception Error Capture Unlock register (meecunlock)

Bits	Field Name	Type	Reset	Description
[31:0]	unlock	WO	0x0	Unlock control for meecaddr. If the address of the first imprecise bus error exception that occurs while the address capture is not locked is captured to the meecaddr register, the meecaddr register is locked. While the register is locked, subsequent errors are not captured. To restart (unlock) the capturing via the meecaddr register, use System Reset or write to this register. When any value is written to this register, capturing via the meecaddr register will be available. When this register is read, "0" is read out.

3.8.3.2.3. mhtpc

mhtpc is a register for setting the PC value of a hart.

The functions of mhtpc are as follows:

- The condition for the transition to Disable or Stop is met when the Hart-level state is Run, the PC value of the hart (the address of the instruction to be executed next) is stored in mhtpc.
- When the Hart-level state is Run, the PC value of the hart is not reflected in mhtpc until the condition for the transition to Disable or Stop is met.
- If the condition for the transition to Run is met when the Hart-level state is Disable or Stop, the value of mhtpc is captured as the PC value of the hart. The hart then starts execution from the PC value.
- If a Resume request is issued when the Hart-level state is DB-Halt and the PEnable value of mhtenable is 0, the PC value of the hart stored in dpc (the address of the instruction to be executed next) is stored in mhtpc.
- If a Halt request is issued when the Hart-level state is Disable or Stop, the PC value of the hart stored in mhtpc (the address of the instruction to be executed next) is stored in dpc.

The expected usage of mhtpc is as follows:

- When the Hart-level state is Disable or Stop, mhtpc can be used to read the PC value (the address of the instruction to be executed next).
- When the Hart-level state is Disable or Stop, mhtpc can be used to write the PC value (the address of the instruction executed by the hart when a run request is issued).

For details of the Hart-level state, see Section 3.3.3.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	PC[31:16]															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	PC[15:2]														PC[1]	—
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO

(*1) cfg_resetvec[31:2]

Table 100. Machine Hart PC Register (mhtpc)

Bits	Field Name	Type	Reset	Description
[31:2]	PC[31:2]	RW	cfg_resetvec[31:2]	PC address[31:2] of hart
[1]	PC[1]	RW	0x0	PC address[1] of hart If the P_RV32C = 0, this bit is RO and always 0.
[0]	-	RO	0x0	Reserved

3.8.3.2.4. mhtenable

mhtenable is a register for controlling the transition of the Disable operation mode of the hart and the other hart operation modes.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	PEnable	Enable
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW

(*1) cfg_boothart

Table 101. Hart Enable Control Status (mhtenable)

Bits	Field Name	Type	Reset	Description
[31:2]	-	RO	0x0	Reserved
[1]	PEnable	RW	cfg_boothart(*1)	This bit holds mhtenable.Enable bit when transiting to DB-Halt mode.
[0]	Enable	RW	cfg_boothart(*1)	This bit enables Hart. When 1 is written while mhtstatus.run=1, Hart's operation state changes to Run mode. When 1 is written while mhtstatus.run=0, Hart's operation state changes to Stop mode. When 0 is written while mhtstatus.run=1or0, Hart's operation state changes to Disable mode. This bit indicates Hart's Enabled/Disabled Status. When read, the following status is read. 1 : Enabled 0 : Disabled

*1 Value of the corresponding hartid bit of cfg_boothart

3.8.3.2.5. mhtstatus

mhtstatus is a register for controlling the start of the execution of the hart.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Debug	Run
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)	(*2)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 102. Hart Run Control Status (mhtstatus)

Bits	Field Name	Type	Reset	Description
[31:2]	-	RO	0x0	Reserved
[1]	Debug	RO	(*1)	This bit indicates Hart's Debug Status. Read: 1: Debug mode (DB-Halt or DB-Run) 0: Not in Debug mode Write: Ignored

Bits	Field Name	Type	Reset	Description
[0]	Run	RW	(*2)	This bit controls Hart’s start/stop operation. Read: 1: Run mode (Run or DB-Run) 0: Not in Run mode Write: 1: Start the hart 0: Stop the hart

*1 1 for Halt-on-reset or 0 in other cases when the reset is released

*2 0 for Halt-on-reset or the value of the corresponding hartid bit of cfg_boothart in other cases when the reset is released

3.8.3.2.6. ml1cachesysctrl

ml1cachesysctrl is a register for controlling whether to enable/disable the L1 instruction cache.
It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC ≥ 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	Cache Enable
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 103. L1 $ System Control (Cache Enable) (ml1cachesysctrll)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0x0	Reserved
[0]	Cache Enable	RW	0x1	L1 Instruction Cache Enable 1 : Enable 0 : Disable

3.8.3.2.7. ml1cacheoperation

ml1cacheoperation is a register for controlling the flushing of the L1 instruction cache.
After the flushing is requested by writing to ml1cacheoperation, the L1 instruction cache will be flushed after the access to the cache by the preceding instruction is completed.
It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC ≥ 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	Flush Operation
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO

Table 104. L1 $ Cache Operation (ml1cacheoperation)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0x0	Reserved
[0]	Flush Operation	WO	0x0	L1 Instruction Flush operation When 1 written, invalidate cache line. When read, 0 is read. 1 : Invalidate cache line 0 : no operation

3.8.3.2.8. ml1cacheprefcfg

ml1cacheprefcfg is a register for controlling the prefetch size (maximum depth) of the L1 instruction cache.
The fetch size of the speculative instruction shown in Section 3.12.2.1 differs depending on the setting value of ml1cacheprefcfg.
For details, see Section 3.12.2.1.
It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC $\geq$ 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	Prefetch Depth		
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW

Table 105. L1 $ Instruction prefetch Control (Max number of prefetches) (ml1cacheprefcfg)

Bits	Field Name	Type	Reset	Description
[31:3]	-	RO	0x0	Reserved
[2:0]	Prefetch Depth	RW	0x2	Max number of Instruction prefetches 001b-111b: Max number of prefetch requests 0: No prefetch

When changing ml1cacheprefcfg, make sure that the instruction fetch is stopped.
The change procedure is as follows.

1. Disable the hart for which ml1cacheprefcfg is to be changed.
   (Write 1 to bit[0] of HARTDIS.)
2. Check that the hart is disabled.
   (Check that 1 is set in bit[0] of STATHTDIS.)
3. Rewrite ml1cacheprefcfg of the hart.

3.8.3.2.9. mbpsysctrl

mbpsysctrl is a register for controlling the branch prediction mechanism.
It is a register that can be read/written when configuration parameter P_BTB is 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	Branch Prediction Enable
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 106. Branch Prediction System Control (mbpsysctrl)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0x0	Reserved
[0]	Branch Prediction Enable	RW	0x1	Branch Prediction Enable 1 : Enable 0 : Disable

When changing the mbpsysctrl setting, make sure that the instruction fetch is stopped.
The change procedure is as follows.

1. Disable the hart for which mbpsysctrl is to be changed. (Write 1 to bit[0] of HARTDIS.)
2. Check that the hart is disabled. (Check that 1 is set in bit[0] of STATHTDIS.)
3. Rewrite mbpsysctrl of the hart.

3.8.3.2.10. mbpoperation

mbpoperation is a register for operating the branch prediction mechanism.
It is a register that can be read/written when configuration parameter P_BTB is 1. Access in any other condition causes a bus error response (DECERR).

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	Flush Operation
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO

Table 107. Branch Prediction Operation (mbpoperation)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0x0	Reserved
[0]	Flush Operation	WO	0x0	BTB Entry Flush Operation 1 : Invalidate BTB entries 0 : No operation

3.8.3.2.11. mhtfetchaddr

mhtfetchaddr is a register for observing the fetch address of the hart.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	ThePC[31:2]															
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	ThePC[31:2]														ThePC[1]	—
Reset:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

x: undefined

Table 108. Hart Fetch address Register (mhtfetchaddr)

Bits	Field Name	Type	Reset	Description
[31:2]	ThePC[31:2]	RO	Undefined	Fetch address[31:2] of Hart
[1]	ThePC[1]	RO	Undefined	Fetch address[1] of Hart RO when P_RV32C = 0. 0 is always read.
[0]	-	RO	0x0	Reserved

3.9. Instruction Set

This section describes the instruction set supported by NS31A.
The NS31A Core supports a 32-bit RISC-V basic instruction set called RV32I and some extended instructions.
See Appendix C for the list of instructions supported by NS31A.

Table 109 shows the supported instruction sets.

Table 109. Instruction Set

Instruction Set
Base Integer (RV32I, RV32E) (*1)
Instruction-Fetch Fence (Zifencei)
Multiply and divide (RV32M)
Atomic operations (RV32A)
CSR instructions (Zicsr)
Single-precision floating point (RV32F) (*2)
Bit manipulation (RV32B) (*3)
Compressed Instructions (RV32C) (*4)

*1 RV32E is enabled when configuration parameter P_RV32E is 1.
*2 Supported only when configuration parameter P_FPU is 1.
*3 Supported only when configuration parameter P_RV32B is 1.
*4 Supported only when configuration parameter P_RV32C is 1.

3.9.1. Base Integer Instruction Set (RV32I, RV32E)

This is compliant with RISC-V specifications.
Select the Base Integer Instruction Set using configuration parameter P_RV32E.
When configured with 0, RISC-V RV32I is supported. When configured with 1, RISC-V RV32E is supported.

3.9.1.1. Implementation-dependent part specifications

Behavior of the load/store instructions when the address is misaligned

NS31A does not support access to misaligned data.

- If misaligned data is accessed via the LH/LHU/LW instruction, a load address misaligned exception will be thrown.
- If misaligned data is accessed via the SH/SW instruction, a store/AMO address misaligned exception will be thrown.

Behavior of the FENCE instruction

Upon execution of a FENCE instruction, NS31A waits until all type memory accesses complete and then starts execution of the subsequent instructions, irrespective of the values in the fm, predecessor, successor, rs1, and rd fields.

3.9.2. Instruction-Fetch Fence (Zifencei)

This is compliant with RISC-V specifications.

For the information about the processor behavior that occurs when the FENCE.I instruction of this instruction set is executed, see Section 3.12.4.

3.9.3. Integer Multiplication and Division (RV32M)

This is compliant with RISC-V specifications.

3.9.4. Atomic Instructions (RV32A)

This is compliant with RISC-V specifications.

3.9.5. Control and Status Register (CSR) Instructions (Zicsr)

This is compliant with RISC-V specifications.

3.9.6. Single-Precision Floating Point (RV32F)

NS31A supports RISC-V RV32F only when configured with configuration parameter P_FPU set to 1. When NS31A is configured with P_FPU set to 0, executing an RV32F instruction causes an Illegal instruction exception to be thrown.

This section describes the unique additional instructions, unique specifications, and implementation-dependent part specifications of NS31A in RV32F.

- Addition of the ns.fexp2.s/ns.flog2.s instructions: Unique additional instructions
- Flushing of subnormal numbers: Unique specifications
- Behavior of the load/store instructions when the address is misaligned: Implementation-dependent part specifications
- Not a Number (NaN) propagation specifications: Implementation-dependent part specifications

3.9.6.1. NS31A unique additional instructions

NS31A supports the following two instructions as unique additional instructions only when configured with both configuration parameters P_FPU and P_FPU_EXP2LOG2 set to 1.

ns.fexp2.s

Single-Precision Floating-Point Base-2 Exponential Instruction

ns.flog2.s

Single-Precision Floating-Point Base-2 Logarithm Instruction

An illegal instruction exception occurs if either of the above unique instruction is executed when configured with either P_FPU or P_FPU_EXP2LOG2 set to 0.

Table 110 shows the encoding of the unique extended instructions.

- Base-2 floating-point exponential instruction (ns.fexp2.s)
  - This instruction performs an exponential calculation for base 2 by using the operand rs1 as an exponent and returns the result to the destination rd. rs1 and rd are single-precision floating-point numbers.
  - The calculation result has an error of up to 1 ulp compared to that obtained with infinite precision.
- Base-2 floating-point logarithm instruction (ns.flog2.s)
  - This instruction performs a logarithm calculation for base 2 by using the operand rs1 as an antilog and returns the result to the destination rd.
    rs1 and rd are single-precision floating-point numbers.
  - The calculation result has an error of up to 1 ulp compared to that obtained with infinite precision.

Table 110. Encoding of the Unique Extended Instructions (RV32F)

Mnemonic	31-27	26-25	24-20	19-15	14-12	11-7	6-0
ns.fexp2.s	01011	00	00100	rs1	rm	rd	0101011
ns.flog2.s	01011	00	00101	rs1	rm	rd	0101011

- For both instructions, the input of subnormal numbers is flushed. In addition, the output of subnormal numbers are flushed, and the Underflow exception flag is set. For details, see Section 3.9.6.2.

Table 111 and Table 112 show the conditions for setting the floating-point exception flags of ns.fexp2.s and ns.flog2.s, respectively.

Table 111. Conditions for Setting the Floating-point Exception Flags of ns.fexp2.s

Floating Point Flags	Conditions to accrue flag
NX: Inexact	Calculated result is not exactly representable.
UF: Underflow	Calculated result underflow.
OF: Overflow	Calculated result overflow.
DZ: Divide by zero	(Never occurs.)
NV: Invalid operation	Input operand: sNaN.

Table 112. Conditions for Setting the Floating-point Exception Flags of ns.flog2.s

Floating Point Flags	Conditions to accrue flag
NX: Inexact	Calculated result is not exactly representable.
UF: Underflow	Calculated result underflow.
OF: Overflow	Calculated result overflow.
DZ: Divide by zero	Input operand: +/- 0
NV: Invalid operation	Input operand: sNaN or negative number.

3.9.6.2. NS31A unique specifications

Flushing of subnormal numbers

In IEEE 754-2008, very small values less than or equal to the minimum normal number are defined as subnormal numbers.
For the unique additional floating-point instructions supported by NS31A, subnormal numbers cannot be handled in compliance with IEEE 754-2008. Calculation can be continued by normalizing (flushing) subnormal numbers to specific values. The flushing of subnormal numbers is applied to Floating-Point Computational Instructions, ns.fexp2.s/ns.flog2.s instruction, defined as unique additional instructions.
In NS31A, the flushing of subnormal numbers in the unique additional instructions is always enabled.

In floating-point operation instructions for NS31A, if a subnormal number is contained in the input operand, or if the calculation result is a subnormal number, the processes that are compliant with IEEE 754-2008 are supported for the instructions specified in RISC-V RV32F.

If the input operand contains a subnormal number, the input operand is always flushed.
If the input operand is flushed, the NX (Inexact) flag is set.
If the calculation result of a floating-point calculation in the unique additional instructions is a subnormal number, the calculation result is always flushed. Whether the input operand has been flushed can be checked that the calculation result has been flushed by referencing the Underflow (UF) bit, which is an exception flag bit defined in RV32F.
According to the RV32F specifications, the UF flag is detected after rounding. Therefore, the flushing of a calculation result is also applied to the result after rounding.

The value where a subnormal number is flushed differs depending on whether the subnormal number is positive or negative and the rounding mode of floating-point calculations. In NS31A, for five rounding modes defined in RV32F, subnormal numbers are normalized to values shown in Table 113.

The Divide by Zero (DZ) or Invalid Operation (NV) exception may occur when a subnormal number in the input operand is normalized to 0. If the DZ/NV exception occurs when the input operand is flushed, the Inexact (NX) flag is not set.

Table 113. Subnormal flush value

Flush value for each Rounding Mode	RNE	RTZ	RDN	RUP	RMM
+Subnormal operand	+0	+0	+0	+0	+0
-Subnormal operand	-0	-0	-0	-0	-0
+Subnormal result	+0	+0	+0	$+2^{min}$	+0
-Subnormal result	-0	-0	-2^{min}	-0	-0

3.9.6.3. Implementation-dependent part specifications

3.9.6.3.1. Behavior of the load/store instructions when the address is misaligned

RISC-V specifies that misaligned floating-point load/store instructions should be processed in the background or trapped. In NS31A, executing an address-misaligned FLW or FSW instruction causes "Load address misaligned" or "Store/AMO address misaligned" exception.

3.9.6.3.2. Not a Number (NaN) propagation

In IEEE 754-2008, a format for representing non-numbers called Not a Number (NaN) is defined as follows: Quiet NaN if the upper bits of the mantissa are set and Signaling NaN if the upper bits of the mantissa are cleared. For floating-point calculation instructions, except fmin/fmax instructions, if the input operand is NaN, the detailed implementation for propagation to calculation results is not defined in RV32F.
As the NaN propagation rule in NS31A, it is not allowed to propagate a NaN of the input operand, and a Canonical NaN (0x7fc00000) defined in RV32F is output as a result.

Table 114 shows the NaN propagation rule.

Table 114. NaN propagation rule

Input Operand	Output Result
Signaling NaN, Quiet NaN, and a number that is not NaN	A Canonical NaN is output.
Only Signaling NaN and Quiet NaN	A Canonical NaN is output.
Only Signaling NaN	A Canonical NaN is output.
Only Quiet NaN	A Canonical NaN is output.
Only Signaling NaN and a number that is not NaN	A Canonical NaN is output.
Only Quiet NaN and a number that is not NaN	A Canonical NaN is output.
Only numbers that are not NaN; if an Invalid Operation (NV) exception occurs	A Canonical NaN is output.

As flag implementation-dependent specifications where NaN is contained in the input operand, the following is specified.

- If NaN is contained in the input operand, Inexact (NX) is not asserted.
- In a group of product-sum instructions (f[n]madd, f[n]msub), if (inf x 0) +/- NaN, Invalid Operation (NV) is asserted.

3.9.7. Bit manipulation extension (RV32B)

NS31A supports RISC-V RV32B only when configured with configuration parameter P_RV32B set to 1. When NS31A is configured with P_RV32B set to 0, executing an RV32B instruction causes an Illegal instruction exception to be thrown. Bit manipulation extention is compliant with RISC-V specifications.

3.9.7.1. Implementation-dependent part specifications

NS31A supports Zba, Zbb and Zbs only.

3.9.8. Compressed Instructions (RV32C)

This is compliant with RISC-V specifications.

3.10. Mutual Exclusion

In NS31A, the mutual exclusion in multi-processing is implemented by using RV32A extended instructions and their functions.
In RV32A, the following two types of operations are provided.

- Atomic Memory Operation (AMO)
- Load-Reserved/Store-Conditional Instructions (LR/SC)

For information about the software processing for mutual exclusion using the AMO and LR/SC instructions, see the RISC-V Instruction Set Manual listed in Reference Documents.

This section describes the application of the RV32A extension to NS31A.

3.10.1. Atomic Memory Operation (AMO)

Table 115 shows the behavior of the AMO instruction for each memory region in NS31A.

Table 115. AMO Region

Memory Region	AXI Configuration			AHB Configuration
	AMO swap	AMO logical	AMO arithmetic	AMO all
ILM Region	✓	✓	✓	-
DLM Region	✓	✓	✓	-
Debug Subsystem Register Region (*1)	✓	✓	slave error	✓ (*4)
Hart Context Register Region (*1)	✓ (*3)	✓ (*3)	slave error	✓ (*3,*4)
ACLINT Register (*1)	✓	✓	slave error	✓ (*4)
CMU Register (*1)	✓	✓	slave error	✓ (*4)
Internal Register Region (*1)	✓	✓	slave error	✓ (*4)
PLIC Register (*1)	✓	✓	slave error	✓ (*4)
External Peripheral Bus Region	✓	✓	slave error	-
System Bus Region (*2)	slave error	slave error	slave error	✓ (*4)

1. In AHB configuration, these regions are accessed from Data Bus Master through external bus and Peripheral Bus Slave. For details of bus requests, see Table 116.
2. In AXI configuration, data accesses of AMO instructions are issued from System Bus Master.
   In AHB configuration, data accesses of AMO instructions are issued from Data Bus Master. For details of bus requests, see Table 116.
3. For a hart context register where control and status registers for the core or instruction cache are mapped, the atomicity cannot be guaranteed if the register is accessed while the target hart is executing a program (Run mode).
4. In AHB configuration, all regions accepts access of AMO instruction from Data Bus Master.

- If a slave error occurs, write to memory and write to the destination register are not performed.
- NS31A does not support access to misaligned data. If misaligned data is accessed, a Store/AMO address misaligned exception will be thrown.

In the case of the AHB configuration, when the NS31A Core executes an AMO instruction, the Data Bus Master Interface issues the bus requests shown in Table 116.
When the NS31A Core executes an AMO instruction, the Data Bus Master Interface issues a lock transfer request using the dbm_hmastlock signal to perform Read-Modify-Write.

Table 116. Bus Requests to the Data Bus Using the AMO Instruction in the AHB Configuration

Signal	AMO Instruction (Read)	AMO Instruction (Write)
dbm_hexcl	0_B	0_B
dbm_hwrite	0_B	1_B
dbm_hmastlock	1_B	1_B
dbm_hsize	010_B	010_B
dbm_hburst	000_B	000_B
dbm_hprot	[3:2] P_DBM_HPROT3_2_FVAL [1] 0-1_B [0] 1_B	[3:2] P_DBM_HPROT3_2_FVAL [1] 0-1_B [0] 1_B

3.10.2. Load-Reserved/Store-Conditional Instructions (LR/SC)

NS31A supports exclusive control using the LR/SC instructions. Table 117 shows the behavior of the LR/SC instructions for each memory region in NS31A.

Table 117. LR/SC Region

Memory Region	AXI Configulation		AHB Configulation	
	LR	SC	LR	SC
ILM Region	✓ (*3)	Always fail (*4)	-	-
DLM Region	✓	✓	-	-
Debug Subsystem Register Region (*1)	✓ (*3)	Always fail (*5)	✓	✓ (*8)
Hart Context Register Region (*1)	✓ (*3)	Bus Error (*6)	✓	✓ (*8)
ACLINT Register (*1)	✓ (*2)	Always fail (*5)	✓	✓ (*8)
CMU Register (*1)	✓ (*2)	Always fail (*5)	✓	✓ (*8)
Internal Register Region (*1)	✓ (*3)	Always fail (*5)	✓	✓ (*8)
PLIC Register (*1)	✓ (*2)	Always fail (*5)	✓	✓ (*8)
External Peripheral Bus Region	✓ (*3)	Always fail (*5)	-	
System Bus Region (*2)	✓	✓ (*7)	✓	✓ (*8)

1. In AHB configuration, these regions are accessed from Data Bus Master through external bus and Peripheral Bus Slave. For details of bus requests, see Table 120.
2. In AXI configuration, data accesses of LR/SC instructions are issued from System Bus Master. For details of bus requests, see Table 119.
   In AHB configuration, data accesses of LR/SC instructions are issued from Data Bus Master. For details of bus requests, see Table 120.
3. It operates as a load operation from a normal register.
4. No store operation to the memory is executed. The failure code (00000001_H) is written to the destination register.
5. A store operation to the memory is executed. The failure code (00000001_H) is written to the destination register.
6. Bus error (SLVERR) occurs. The failure code (00000001_H) is written to the destination register.
7. The success/failure of an SC instruction depends on the write response type (the value of BRESP) received from the System Bus Master.
8. The success/failure of an SC instruction depends on the write response type (the value of dbm_hresp, dbm_hexokay) received from the Data Bus Master.

3.10.2.1. Detailed behavior of the LR/SC instructions

In the memory region supporting the exclusion control shown in Table 117, execute the Load-Reserved (LR) instruction to reserve an address, and then execute the Store-Conditional (SC) instruction at the same address.
If the exclusion has succeeded, write 00000000_H to the destination register as the execution result of the SC instruction. If the exclusion has failed, write 00000001_H to the destination register. From a single hart, only a single address can be reserved. Even when the memory region differs, multiple addresses cannot be reserved at the same time.
Reservation by the LR instruction can be canceled by the behavior of the hart that has executed the LR instruction shown

in Table 118.

Table 118. Conditions for Canceling Reservation

Category	Behavior
Instruction	・Execute the following instructions between the execution of the LR instruction and the execution of the SC instruction. (For the instruction sets supported in NS31A, see Table 109.) - MRET instruction - MNRET instruction
Interrupt	Interrupts shown in Table 19
Exception	Exceptions shown in Table 16
Operating status	The Core-level status shown in Figure 5 changes to Reset/Core Inactive. The Hart-level status shown in Figure 6 changes to Reset/Disable.

When a program including LR and SC instructions is debugged by using a debugger, if a breakpoint is set between the LR instruction and the SC instruction or if the stepwise execution is performed between the LR and SC instructions, the Hart-level status changes to DB-Halt, but explicit reservation cancellation by hardware is not performed.

In the case of the AXI configuration

When the NS31A Core executes an LR/SC instruction that accesses the System Bus Region, the System Bus Master Interface issues a bus request shown in Table 119.

Table 119. Bus Requests to the System Bus by the LR/SC (In the case of the AXI configuration)

Signal	LR Instruction (AR channel)	SC Instruction (AW channel)
AxLOCK	1_B	1_B
AxSIZE	010_B	010_B
AxBURST	01_B	01_B
AxLEN	00_H	00_H
AxCACHE	P_SBM_AXCACHE_FVAL	P_SBM_AXCACHE_FVAL
AxPROT	[2] 0_B [1] P_SBM_AXPROT1_FVAL [0] 0-1_B	[2] 0_B [1] P_SBM_AXPROT1_FVAL [0] 0-1_B

In the case of the AHB configuration

When the NS31A Core executes an LR/ SC instruction, the Data Bus Master Interface issues the bus requests shown in Table 120.
When the NS31A Core executes an LR/SC instruction, the Data Bus Master Interface issues an exclusive access bus request using the dbm_hexcl signal.

Table 120. Bus Requests to the Data Bus Using the LR/SC Instruction (in the Case of the AHB Configuration)

Signal	LR Instruction	SC Instruction
dbm_hexcl	1_B	1_B
dbm_hwrite	0_B	1_B
dbm_hmastlock	0_B	0_B
dbm_hsize	010_B	010_B
dbm_hburst	000_B	000_B
dbm_hprot	[3:2] P_DBM_HPROT3_2_FVAL [1] 0-1_B [0] 1_B	[3:2] P_DBM_HPROT3_2_FVAL [1] 0-1_B [0] 1_B

The Data Bus Master Interface outputs no bus master ID in the AHB configuration.
Be sure to add a desired ID from the outside of NS31A if an external AHB requires the bus master ID.

3.10.3. Exclusive access support by an external AXI master

In the case of the AXI configuration, NS31A supports exclusive access from an external AXI master, as well as exclusive control between the NS31A Core and the external AXI master.
For exclusive access from the Front Port, the requests shown in Table 121 are supported.

Table 121. Exclusive Access from the Front Port

Signal	DLM
AxLOCK	1_B
AxSIZE	010_B
AxBURST	01_B
AxLEN	00_H
AxCACHE	Don't care
AxPROT	Don't care

Exclusive access other than those shown in Table 121 causes a bus error response (SLVERR).

In the case of the AHB configuration, exclusive access from an external APB master to the Peripheral Bus Slave is not supported.

3.10.4. Exclusive monitor

In the case of the AXI configuration, NS31A has an exclusive monitor in each memory of the Internal RAM Region, which makes it possible to implement exclusive control by means of LR/SC instructions from the NS31A Core and exclusive access from an external AXI master.

In the case of the AHB configuration, the memories of the Internal RAM Region are not mounted in NS31A.

3.10.4.1. Number of exclusive monitor entries

Table 122 shows the number of exclusive monitor entries included in each memory of the Internal RAM Region.
Number of addresses that can be reserved for exclusive access from a single hart or the Front Port with the same AxID is only one. If the second address is reserved, the address that was reserved earlier is canceled.

Table 122. Number of Exclusive Monitor Entries

RAM	# of Entries
DLM	2

3.10.4.2. Conditions for canceling exclusive monitor reservation

The reservation of each exclusive monitor entry is canceled if any of the following conditions is met.

- Store-Conditional is executed by the master that made the reservation.
- Store/AMO instructions by the hart of the NS31A Core that match the address alignment shown in Table 123

- Store access by the external master that matches the address alignment shown in Table 123

Table 123. Store Address Alignment Subject to Entry Clearance

RAM	Master	Store Address Alignment
DLM	All	4 Bytes

When data whose size is not a power of 2 is transferred from the Front Port Bus Slave via the burst write transfer, it is converted to a size of a power of 2 on the internal bus.
Dummy write data complements the shortage, but it is not written to DLM.
If the burst write address region or dummy write region shown above contains an address alignment shown in Table 123, the exclusive monitor reservation is canceled.

3.10.4.3. Exclusive monitor entry access

Exclusive monitor entries can be accessed directly for software debugging.
For the entry registers, see Table 124.

Table 124. Exclusive Monitor Entry Registers

Memory	Register Category	Register	Register Descriptions
DLM	EXMON	EXCLMONVALIDBITENTRYn	Section 3.8.3.1.1
		EXCLMONADDRESSENTRYn	Section 3.8.3.1.2
		EXCLMONIDENTRYn	Section 3.8.3.1.3
		EXCLMONSIDEBANDENTRYn	Section 3.8.3.1.4

3.11. Branch Prediction

When configured with P_BTB set to 1, a branch prediction mechanism using the BTB is mounted in NS31A and is enabled immediately after the reset is released.
The branch prediction mechanism can be disabled by using mbpsysctrl.
The BTB entries can be cleared by executing the FENCE.I instruction or using mbpoperation.
The BTB entry data can be accessed from the test interface.

For details about how to use each register, see the corresponding section shown below.

- mbpsysctrl (Section 3.8.3.2.9)
- mbpoperation (Section 3.8.3.2.10)

For details of the test interface, see Section 3.11.1.

These functions are not available when configured with P_BTB set to 0.

3.11.1. BTB entry Test interface

BTB entry data can be accessed directly by using BTBACTRL, BTBATGT, and BTBADAT.
For details of the control register for accessing BTB entry data, see the corresponding section shown below.

- BTBACTRL (Section 3.8.3.1.11)
- BTBATGT (Section 3.8.3.1.12)
- BTBADAT (Section 3.8.3.1.13)

Access BTB entry data by using the following procedures.

Procedure for writing BTB entry data

1. Set the access target and its address in the BTBATGT register.
2. Set the data to be written in the BTBADAT register.
3. Write 2 or 3 to the RAC[1:0] bits of the BTBACTRL register.

Procedure for reading BTB entry data

1. Set the access target and its address in the BTBATGT register.
2. Write 1 to the RAC[1:0] bits of the BTBACTRL register.
3. Read the data from the BTBADAT register.

3.12. Usage Notes

This section describes precautions for the behavior of the processor under the following conditions.
There are the following items.

- Memory consistency model
- Amount of look-ahead in speculative access to a memory
- Behavior if an error occurs due to memory access
- Behavior if the FENCE.I instruction is executed
- Behavior when PMP setting is changed
- Minimum region granularity of instruction fetch for PMP
- Optimization of the integer multiplier and divider

3.12.1. Memory Consistency Model

In the RISC-V ISA, RISC-V Weak Memory Ordering (RVWMO) is defined as the memory consistency model, and a memory access instruction executed in a single hart is monitored as is executed in the order of the program codes. At this time, however, the memory access instruction in a hart might be monitored in the order different from that in which the instruction is actually executed, from a view point of the other harts and bus slaves. Also, the instruction subsequent to a memory access instruction in a hart can be executed without waiting for completion of the preceding memory access instruction, unless there is no data-dependent relationship between the instructions.

The value read by a CSR instruction reflects the execution result of the previous instruction, and the value to be written by a CSR instruction is reflected in the execution result of the subsequent instruction. Note, however, that this does not mean that the effect resulted from a memory access is always reflected in the CSR immediately.

For details of the memory consistency mode in RISC-V ISA, see the RISC-V Instruction Set Manual listed in Reference Documents.

In NS31A, all memory accesses including an access to a device are executed in the order of instructions, but the subsequent memory access may start before the preceding memory access completes. In addition, unless otherwise specified in the RISC-V ISA and in this document, an instruction other than the memory access instruction may be executed before the preceding memory access completes. If it is required to wait for completion of the preceding memory access, therefore, use a FENCE instruction.

When a memory access is used to configure the CMU settings, a part of the specified values are applied to the CSR. Note, however, that this does not guarantee the CMU settings being applied immediately to the CSR. If the specified values need to be applied certainly, be sure to execute a CSR read instruction to confirm that the values are applied and then start execution of the subsequent instruction. The order between a memory access instruction and an instruction fetch is not guaranteed.

Execute a FENCE.I instruction if it is required to guarantee the order between a memory access instruction and an instruction fetch.

For details of the behavior of the FENCE.I instruction, see Section 3.12.4.

3.12.2. Amount of look-ahead in speculative access to a memory

3.12.2.1. Instruction fetch

Since the instruction fetch by NS31A is executed speculatively, there is a case where up to 256 preceding bytes (*1) are loaded from the address of a branch instruction. Speculative instruction fetch is not performed when P_CACHE_SIZE_IC is set to 0.
This speculative fetch size differs depending on the memory region where instruction codes are allocated.
When implementing ECC RAM, etc. outside, it is needed to take measures, such as the initialization of regions including the region where speculative fetch may be performed when instruction codes are allocated, to suppress the occurrence of unintended ECC errors, etc.

[*1] If 7 is set as the maximum instruction prefetch size (depth) (For the setting register for the maximum instruction prefetch size, see the Prefetch Depth field in Section 3.8.3.2.8.)

The instruction fetch address range output by speculative instruction fetch differs depending on the maximum instruction prefetch size.

- When Prefetch Depth = 0
  - All the addresses up to the "end of the address region aligned with 32 bytes including the branch instruction + 32 bytes" are output.
- When Prefetch Depth >= 1
  - All the addresses up to the "end of the address region aligned with 32 bytes including the branch instruction + (32 bytes * Prefetch Depth)" are output.

Table 125 shows the maximum speculative instruction fetch size in each memory region when Prefetch Depth is 0.

Table 125. Speculative Instruction Fetch Size (Prefeth Depth=0)

Region	Instruction Fetch Size
(All regions)	64 bytes

3.12.3. Behavior if an error occurs due to memory access

Table 126 shows the behavior performed if an error occurs due to memory access.

Table 126. Memory Access Error behavior

Access	Error Type	Exception cause (m[n]cause)	m[n]epc	mtval	Comment
I-Fetch	Bus Response Error (Slave Error / Decode Error) (*1)	Fetch Bus Error NMI	Next PC (*2)	-	Precise, NMI
	PMP Error	Instruction access fault exception	Current PC	Fetch Address	Precise exception
	Uncorrectable ECC Errors	Fetch Bus Error NMI	Next PC	-	Precise, NMI
Load	Bus Response Error (Slave Error / Decode Error) (*1)	Non-blocking load bus error NMI	Next PC	-	Imprecise, NMI
	PMP Error	Load access fault exception	Current PC	Access Address	Precise exception
	Misaligned Accesses	Load address misaligned exception	Current PC	Access Address	Precise exception
	Uncorrectable ECC Errors	Non-blocking load bus error NMI	Next PC	-	Imprecise, NMI
Store	Bus Response Error (Slave Error / Decode Error) (*1)	Non-blocking store bus error NMI	Next PC	-	Imprecise, NMI
	PMP Error	Store/AMO access fault exception	Current PC	Access Address	Precise exception
	Misaligned Accesses	Store/AMO address fault exception	Current PC	Access Address	Precise exception
	Uncorrectable ECC Errors	Non-blocking store bus error NMI	Next PC	-	Imprecise, NMI

1. In the case of the AHB configuration, replace "Bus Response Error (Slave Error / Decode Error)" with "Bus Response Error".
2. During an instruction fetch from the System Bus Master (32-bit data width), if an error response occurs only in a specific beat of the burst read response, a bus response error may occur even when the instruction fetch is executed for a region where no error response occurs.
   This is because the whole 64-bit aligned region, including the region where an error response occurs, is regarded to be in an error state in this case.

3.12.4. Behavior if the FENCE.I instruction is executed

The FENCE.I instruction can be used to achieve synchronization between memory write and instruction fetch.

If the FENCE.I instruction is executed, the following behaviors are performed.

- The previously fetched instruction codes after the FENCE.I instruction are discarded, and fetch is executed again.
- All BTB entries are cleared.
  (Only when configuration parameter P_BTB is 1)
- The cache data in the L1 instruction cache is cleared.
  (Only when configuration parameter P_CACHE_SIZE_IC $\geq$ 1)

If any instruction code is rewritten, execute the FENCE.I instruction to ensure that the rewriting result is reflected.
A concrete example of the execution procedure is shown below.

1. Rewrite the instruction codes.
2. Execute the FENCE.I instruction.

The above procedure is not necessary for instruction fetch from a program buffer in the Debug mode.

When an instruction code on the memory is rewritten to the EBREAK instruction in Debug mode to set a breakpoint, execute the FENCE.I instruction in Debug mode to ensure that the rewriting result is reflected.

3.12.5. Behavior when PMP setting is changed

When a PMP entry is locked (pmpncfg.L=1) using the CSR instruction in Machine mode, the PMP access permission determination may not be performed immediately after the subsequent instruction execution. The access permission is always determined, however, when an instruction subsequent to the FENCE.I instruction is executed or when an exception/interrupt program upon acknowledgment of an exception/interrupt is executed after the entry lock.

In Machine mode, if it is required to apply the PMP setting to the instruction fetch immediately after the entry lock, execute the FENCE.I instruction immediately after the CSR instruction that specifies the entry lock. For details of the behavior of the FENCE.I instruction, see Section 3.12.4.

For data access in Machine mode, however, it is not necessary to consider the timing to apply the PMP setting described above, because the PMP access permission determination is performed for the instruction immediately after the entry lock.

In User mode, the PMP setting changed in Machine mode is applied to the instruction fetch and data access for the instruction next to the mret instruction.

3.12.6. Minimum region granularity of instruction fetch for PMP

For an entry used to determine the permission of instruction fetch access, the granularity must be set to 32 bytes (256 bits) or greater. If not, an access violation may not be detected.
The following shows notes on the PMP setting for instruction fetch.

- When pmpncfg.A is NAPOT and the granularity is 32 bytes, address[4:2] of PMPADDRn must be set to 011b. (n = 0 to 15)
- When pmpncfg.A is TOR, address[4:2] of PMPADDRm and PMPADDRn must be set to 000b. (n = 1 to 15, m = n - 1)
- Do not specify NA4 for pmpncfg.A of the entry that is used to determine whether to permit an instruction fetch access.

For an entry used to determine the permission of data access only, the granularity can be set to less than 32 bytes (256 bits).

3.12.7. Optimization of the integer multiplier and divider

NS31A has an integer multiplier and an integer divider to execute RV32M instructions. You can select whether to increase computing performance or reduce the number of gates, using the relevant configuration parameters.

- Configuration option of the integer multiplier (Can be changed using configuration parameter P_MUL_SMALL.)
  - Fast multiplier (high-performance, P_MUL_SMALL=0)
    latency = 2 clk cycle
  - Small multiplier (area-efficient, P_MUL_SMALL=1)
    latency = 16-17 clk cycle
- Configuration option of the integer divider (Can be changed using configuration parameter P_DIV_SMALL.)
  - Fast divider (high-performance, P_DIV_SMALL=0)
    latency = 4-20 clk cycle
  - Small divider (area-efficiency, P_DIV_SMALL=1)
    latency = 1-38 clk cycle

Chapter 4. Bus Interface

This chapter describes the external bus interfaces of NS31A.

4.1. Interfaces

NS31A has the external bus interfaces listed below.
Each external bus interface is compliant with the AMBA AXI4, AHB5 lite, and APB4 specifications.
For the detailed specifications of each AMBA bus protocol, see Reference Documents.

- AXI configuration
    - System Bus Master Interface (AXI4)
    - External Peripheral Bus Master (APB4)
    - Front Port Bus Slave Interface (AXI4)
- AHB configuration
    - Instruction Bus Master Interface (AHB5 lite)
    - Data Bus Master Interface (AHB5 lite)
    - Peripheral Bus Slave Interface (APB4)

The following sections describe the individual bus interfaces in detail. They also describe how the special transfers shown below are supported.
These special transfers are available for the AXI bus. The AHB and APB buses do not support them.

Narrow burst transfer

Transfer with AxLEN is 1 or greater (burst transfer), in which the maximum beat transfer size specified by AxSIZE is smaller than the bus data bit width.

Unaligned transfer

Transfer with AxLEN is 0, in which AxADDR is not aligned to the size specified by AxSIZE. Otherwise, transfer with AxLEN is 1 or greater (burst transfer), in which the transfer start address specified by AxADDR is not aligned to the bus data bit width.

Sparse write byte-strobe

Transfer in which write to the specific regions is disabled when WSTRB is set to “0” for some byte lanes that are used during write transaction. Note that the transfer is categorized as “Zero write byte strobe” if WSTRB is set to “0” for all byte lanes during transaction.

Zero write byte-strobe

Transfer in which write is disabled when WSTRB is set to “0” for all byte lanes that are used during write transaction.

Read data interleaving

Transfer in which a read response with a different ID exists between the start and end of a read burst response with a certain ID during read transaction.

4.1.1. System Bus Master

The System Bus Master is available in the AXI configuration. It is used as the AXI4 Master interface for external access. The Bus Master in NS31A accesses external RAM and other resources via the System Bus Master interface.
The System Bus Master interface is compliant with the AXI4 protocol. For the detailed specifications of the System Bus Master interface, see Table 127.

Table 127. System Bus Master Spec

Item		System Bus Master Spec	
Master/Slave		Master	
Protocol		AXI4	
Channel		AR/R	AW/W/B
Bit width	Address	32	
	Data	32	
Maximum burst length	Number of beats	8	1
	Number of bytes	32	4
Maximum number of outstanding transactions		1	1
Transaction ID bit width		5	
Burst type	FIXED	Not output	
	INCR	Fixed to INCR (2'b01)	
	WRAP	Not output	
Protection type signal	AxPROT[2]	Supported	
	AxPROT[1]	Fixed to the P_SBM_AXPROT1_FVAL setting value	
	AxPROT[0]	Supported [*1] [*2]	
Cache type signal	AxCACHE[3:0]	Fixed to the P_SBM_AXCACHE_FVAL[3:0] setting value	
Lock signal	AxLOCK[0] (Exclusive)	Supported	
Other signals	AxREGION	Not implemented	
	AxQOS	Not implemented	
Special transfer	Narrow burst transfer	Not output	
	Unaligned transfer	Not output	
	Sparse write byte-strobe	-	Not output
	Zero write byte-strobe	-	Not output
	Read data interleaving	Does not occurred [*3]	-

*1 For access in D/M-mode and DBG-initiated access, 1 is output. For access in U-mode, 0 is output. When the mode is M-mode and mstatus.MPRV is 1, the value equivalent to the mode setting in mstatus.MPP is output.
*2 When P_BUS_PROT0_PRIV configration parameter is 0, the system bus access is fixed to privileged.
*3 Since the maximum number of outstanding transactions is 1, no interleave occurs. However, the transaction ID is not fixed.

4.1.2. External Peripheral Bus Master

The External Peripheral Bus Master is available in the AXI configuration. It is used as the APB Master interface for external access.
The bus master in NS31A accesses peripheral modules and other registers via the External Peripheral Bus Master interface.
The External Peripheral Bus Master interface is compliant with the 32-bit APB4.
For the detailed specifications of the External Peripheral Bus Master interface, see Table 128.

Table 128. External Peripheral Bus Master Spec

Item		External Peripheral Bus Master Spec
Master/Slave		Master
Protocol		APB4
Bit width	Data	32
Protection type signal	PPROT[2:0]	Not supported (not implemented)
Strobe signal	PSTRB[3:0]	Supported
Error signal	PSLVERR	Supported

4.1.3. Front Port Bus Slave

The Front Port Bus Slave is available in the AXI configuration. It is used as interface for internal RAM access.
The bus master outside NS31A accesses internal RAM and others via the Front Port Bus Slave interface.
The Front Port Bus Slave interface is compliant with the AXI4 protocol.
For the detailed specifications of the Front Port Bus Slave interface, see Table 129.

Table 129. Front Port Bus Slave Spec

Item		Front Port Bus Slave Spec	
Master/Slave		Slave	
Protocol		AXI4	
Channel		AR/R	AW/W/B
Bit width	Address	32	
	Data	32	
Maximum burst length (*1)	Number of beats	256	256
	Number of bytes	1024	1024
Maximum number of outstanding transactions		11	11
Transaction ID bit width		P_FBS_AXID_WIDTH setting value	
Burst type	FIXED	Error response (SLVERR)	
	INCR	Supported	
	WRAP	Error response (SLVERR)	
Protection type signal	[2]Instruction	Supported	
	[1]Non-secure	Ignored	
	[0]Privileged	Supported (*4)	

Item		Front Port Bus Slave Spec	
Cache type signal	[3]Write-allocate	Ignored	
	[2]Read-allocate	Ignored	
	[1]Modifiable	Ignored	
	[0]Bufferable	Ignored	
Lock signal	[0]Exclusive	Supported	
Other signals	AxREGION	Not implemented	
	AxQOS	Not implemented	
Special transfer	Narrow burst transfer	Error response (SLVERR)	
	Unaligned transfer	Error response (SLVERR)	
	Sparse write byte-strobe	-	Supported (*2)
	Zero write byte-strobe	-	Supported (*3)
	Read data interleaving	Output	-

1. The burst transfer is valid for the ILM and DLM regions only. Since the burst transfer is not valid for other regions, the maximum number of beats is 1 and the number of bytes is 4 bytes for them. Therefore, an error response (SLVERR) occurs if it is requested for such a region to transfer data exceeding the maximum limit.
2. Sparse write byte-strobe transfer to a register is supported only when the content of the access to the 32-bit region corresponding to the register matches the access size supported for the register. An error response (SVLERR or DECERR) is returned for any other access.
   Note that an access may fail if its destination is a register where idempotence is not guaranteed.
3. Zero write byte-strobe transfer to a register causes an error response (SLVERR or DECERR).
4. The front port bus slave access is fixed to "Privileged" by setting configuration parameter P_BUS_PROT0_PRIV to 0.

4.1.4. Instruction Bus Master Interface

The Instruction Bus Master is available in the AHB configuration. It is used as the AHB5 lite Master interface for instruction access. It is compliant with the AHB5 lite protocol.
NS31A fetches instructions from the Instruction Bus Master. When an instruction cache is mounted and enabled, 32-bit, 8-beat burst transfers are performed. When an instruction cache is disabled or not mounted, 32-bit, single-beat transfers are performed.
For the detailed specifications of the Instruction Bus Master interface, see Table 130.

Table 130. Instruction Bus Master Spec

Item		Instruction Bus Master spec
Master/Slave		Master
Protocol		AHB5 lite
Bit width	Address	32
	Data	32
Maximum burst length	Number of beats	8
	Number of bytes	32
Burst type	HBURST	SINGLE (3'b000) INCR8 (3'b101)
Protection type signal	HPROT[0]	Fixed to 0 (Inst)
	HPROT[1]	Supported [*1]
	HPROT[2]	P_IBM_HPROT_3_2_FVAL Fixed to the setting value
	HPROT[3]	
Strobe signal	HWSTRB	Not supported
Lock signal	HMASTLOCK	Not supported
Other signal	HEXCL	Not supported
	HEXOKAY	Not supported

*1 When configuration parameter P_BUS_PROT0_PRIV is set to 0, the access is fixed to Privileged.

4.1.5. Data Bus Master Interface

The Data Bus Master is available in the AHB configuration. It is used as the AHB5 lite Master interface for data access.
NS31A makes data access to all addresses from the Data Bus Master.
32-bit, single-beat transfers are performed according to the AHB5 lite protocol.
For the detailed specifications of the Data Bus Master interface, see Table 131.

Table 131. Data Bus Master Spec

Item		Data Bus Master spec
Master/Slave		Master
Protocol		AHB5 lite
Bit width	Address	32
	Data	32
Maximum burst length	Number of beats	1
	Number of bytes	4
Burst type	HBURST	Fixed to SINGLE (3’b000)
Protection type signal	HPROT[0]	Fixed to 1 (Data)
	HPROT[1]	Supported *1
	HPROT[2]	P_DBM_HPROT_3_2_FVAL Fixed to the setting value
	HPROT[3]	
Strobe signal	HWSTRB	Not supported
Lock signal	HMASTLOCK	Supported
Other signal	HEXCL	Supported
	HEXOKAY	Supported

*1 When configuration parameter P_BUS_PROT0_PRIV is set to 0, the access is fixed to Privileged.

4.1.6. Peripheral Bus Slave Interface

The Peripheral Bus Slave is available in the AHB configuration. It is used as an interface for internal register access.
A master outside NS31A accesses internal registers via the Peripheral Bus Slave.
The Peripheral Bus Slave is compliant with the 32-bit APB4.
For the detailed specifications of the Peripheral Bus Slave interface, see Table 132.

Table 132. Peripheral Bus Slave Spec

Item		Peripheral Bus Slave spec
Master/Slave		Slave
Protocol		APB4
Bit width	Data	32
Protection type signal	PPROT[0]	Supported *1
	PPROT[1]	Ignored
	PPROT[2]	Supported
Strobe signal	PSTRB[3:0]	Supported
Ready signal	PREADY	Supported
Error signal	PSLVERR	Supported

*1 When configuration parameter P_BUS_PROT0_PRIV is set to 0, the access is fixed to Privileged.

4.2. Features

The following describes additional functions in bus interfaces.

4.2.1. Sideband Information

NS31A attaches Sideband Information to each external bus interface. Table 133 shows the meaning of each sideband signal.

Table 133. Sideband Definition

Category	Signal Name	Description
Master Identification	DBG	Identification signal for the access issued by debug. If DBG bus master or Core bus master under debug mode issue an access, this signal is asserted to 1’b1, otherwise 1’b0.

4.2.1.1. System Bus Master Sideband

The implementation of the Sideband signals of the System Bus Master interface is shown in Table 134.

Table 134. System Bus Master Sideband Information

Bit	ARUSER	RUSER	AWUSER	WUSER	BUSER
0	DBG	No pin	DBG	No pin	No pin

Like other AR/AW channel signals, the value of the Sideband signal does not change until the handshake succeeds after AxVALID is asserted.

4.2.1.2. Front Port Bus Slave Sideband

The implementation of the Sideband signals of the Front Port Bus Slave interface is shown in Table 135.

Table 135. Front Port Bus Slave Sideband Information

Bit	ARUSER	RUSER	AWUSER	WUSER	BUSER
0	DBG	No pin	DBG	No pin	No pin

Like other AR/AW channel signals, do not change the value of the Sideband signal until the handshake succeeds after AxVALID is asserted. If the value is changed, a function that uses the Sideband signal may not work normally.

4.2.1.3. Instruction Bus Master Sideband

The implementation of the Sideband signals of the Instruction Bus Master interface is shown in Table 136.

Table 136. Instruction Bus Master Sideband Information

Bit	HAUSER	HWUSER	HRUSER	HBUSER
0	DBG	No pin	No pin	No pin

The value of the Sideband signal does not change until the address phase is complete.

4.2.1.4. Data Bus Master Sideband

The implementation of the Sideband signals of the Data Bus Master interface is shown in Table 137.

Table 137. Data Bus Master Sideband Information

Bit	HAUSER	HWUSER	HRUSER	HBUSER
0	DBG	No pin	No pin	No pin

The value of the Sideband signal does not change until the address phase is complete.

Chapter 5. System Control

This chapter describes the system control of NS31A.

5.1. Clock

Table 138 shows the list of clocks used by NS31A.

Table 138. List of NS31A Clocks

#	NS31A Clock	Description
1	clk	System clock
2	clk_sf	System clock for the functional safety redundant circuit
3	jtag_tck	JTAG clock

Table 139 shows the synchronization relationship of clocks listed in Table 138.
Be sure to supply clk and clk_sf at the same time.

Table 139. NS31A Clock Synchronization Relationship

	clk	clk_sf	jtag_tck
clk	-	Identical	Async
clk_sf	Identical	-	Async
jtag_tck	Async	Async	-

5.2. Reset

Table 140 shows the reset input signals of NS31A and reset targets.

Table 140. NS31A Reset Specifications

#	NS31A Input Pin	Reset Target			
		NS31A CXW	DBG	DBG JTAG TAP	Others
0	jtag_trst_n	-	-	✓	-
1	rst_pon	-	✓	✓	-
2	rst_sys	✓	✓*1	-	✓

*1: Some parts of DBG are initialized by rst_sys.

rst_pon and rst_sys are synchronous reset signals.
To complete the logical initialization by reset, it is needed to supply clk and clk_sf (when DCLS is used) at least 10clk cycles while the synchronous reset signals are kept being asserted.
For details of the reset timing, see Section 5.2.1 and Section 5.2.2.

Of these reset signals, signals other than jtag_trst_n are distributed to each unit in NS31A after the retiming of the signals is performed by the CMU.
For the reset signals in NS31A, see Chapter 6.

5.2.1. Power-on Reset

When turning on the power of NS31A, assert all the reset signals shown in Table 140, except jtag_trst_n, and control each reset signal according to the prescribed intervals shown in Table 141. When asserting the signals, assert rst_pon and rst_sys in the same clk cycle or assert rst_sys first.

Table 141. Release Timing of NS31A Power on Reset

#	Target	Minimum Interval
1	From the driving of clocks to the negation of rst_pon	10 clk cycles or more
2	From the negation of rst_pon to the negation of rst_sys	1 clk cycle or more

The value of the signal output from NS31A are not finalized until the logical initialization by the Power-on Reset completes after NS31A is turned on.

At the time of system reset, the system can be reset while avoiding a bus deadlock outside NS31A, by executing the system reset via the System Stop state. For details, see Section 5.3.3.

Do not change the jtag_tck and jtag_trst_n signal values when negating rst_pon.
For the detailed timing, see NS31A Integration Manual.

5.2.2. System Reset

When performing system reset for NS31A, control each reset signal according to the prescribed intervals shown in Table 142.

Table 142. NS31A System Reset Timing

#	Target	Minimum Interval
1	From the assertion of rst_sys to the negation of rst_sys	10 clk cycles or more

To avoid a bus deadlock outside NS31A while performing system reset, it is necessary to ensure that the bus transaction executed on the NS31A Front Port Bus Slave is completed.

5.2.3. JTAG Reset

When accessing the DBG unit from the JTAG of NS31A, execute the reset while keeping jtag_trst_n to low or jtag_tms to high for 5 jtag_tck cycles.

- The JTAG specifications are compliant with IEEE 1149.1.
- jtag_trst_n is an asynchronous reset signal that is active in the low state.
- The reset range is the JTAG TAP controller and IR register installed in the DBG units (DTM).

Precautions about JTAG reset

When resetting JTAG TAP by using jtag_trst_n, it is needed to keep jtag_tms to 1 while jtag_trst_n is changed from 0 to 1 (when releasing the reset) in order to ensure that the TAP state at the release of the reset is Test-Logic-Reset.
If jtag_tms is not kept to 1, the state of JTAG TAP may be unstable when releasing the reset.

Stop jtag_tck when asserting or negating jtag_trst_n.
For the detailed timing, see NS31A Integration Manual.

5.3. System-level State Control

NS31A can change the System-level State when requested from an external pin.
For details of this function, see Section 3.3.

The following shows the timing of the transition of System-level State.
Each State control signal is required to follow the conditions described in Section 5.3.1, Section 5.3.2, and Section 5.3.3.
Do not assert ncc_mststop_req, ncc_sysstop_req, and ncc_active_req at the same time.

5.3.1. Master Stop to Active

This section shows the timing when the state returns directly to the Active state via the Master Stop state.

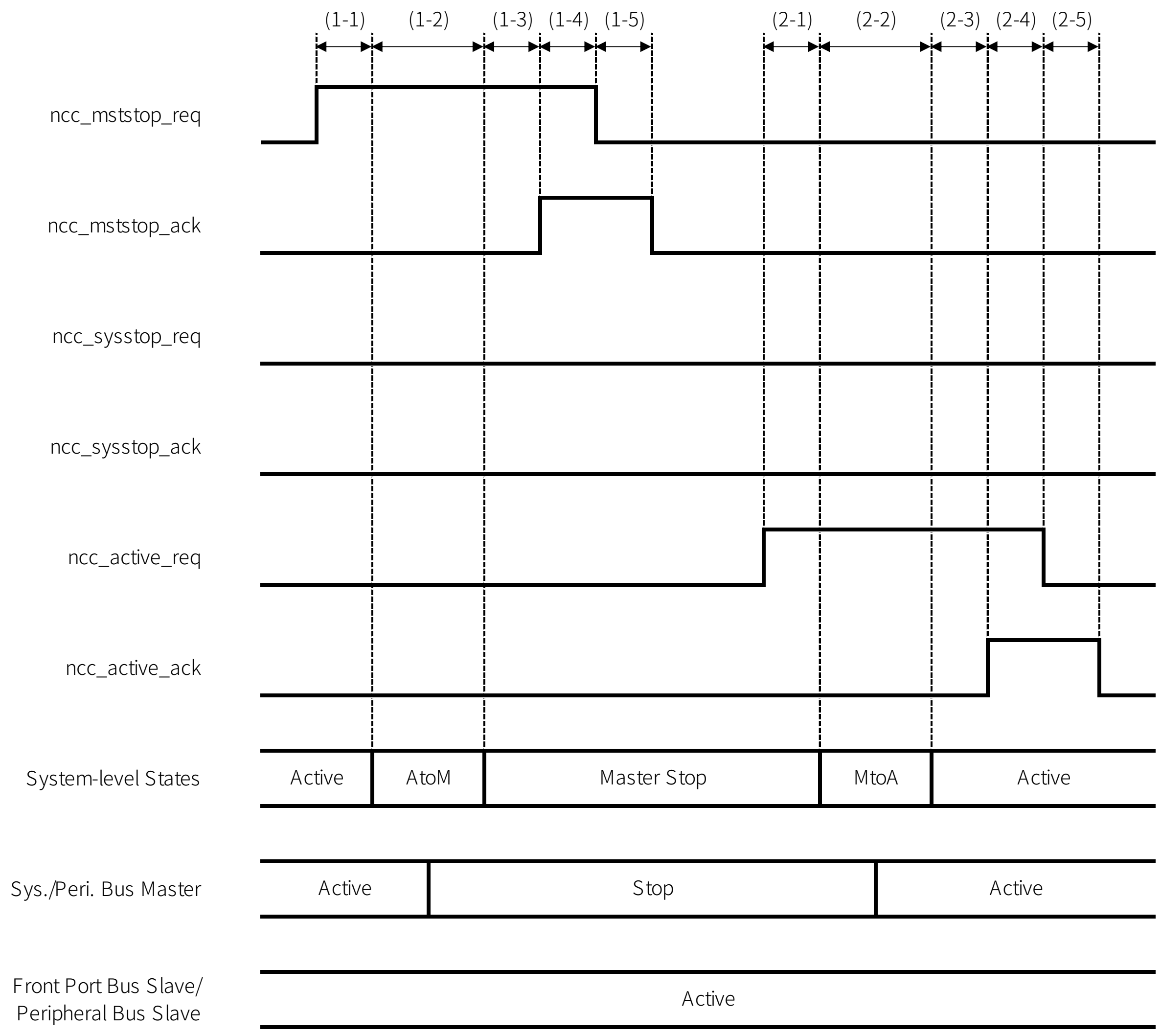


Figure 7. System-level State Transition Timing (Master Stop to Active)

Table 143. System-level State Signal Timing (Master Stop to Active)

Section	Interval
(1-1)	1+ clk cycle
(1-2)	Depends on the bus transaction status
(1-3)	1+ clk cycle
(1-4)	Min 1 clk cycle (external request specification)
(1-5)	1+ clk cycle
(2-1)	1+ clk cycle
(2-2)	1+ clk cycle
(2-3)	1+ clk cycle
(2-4)	Min 1 clk cycle (external request specification)
(2-5)	1+ clk cycle

5.3.2. System Stop to Active

This section shows the timing when the state returns directly to the Active state via the System Stop state.

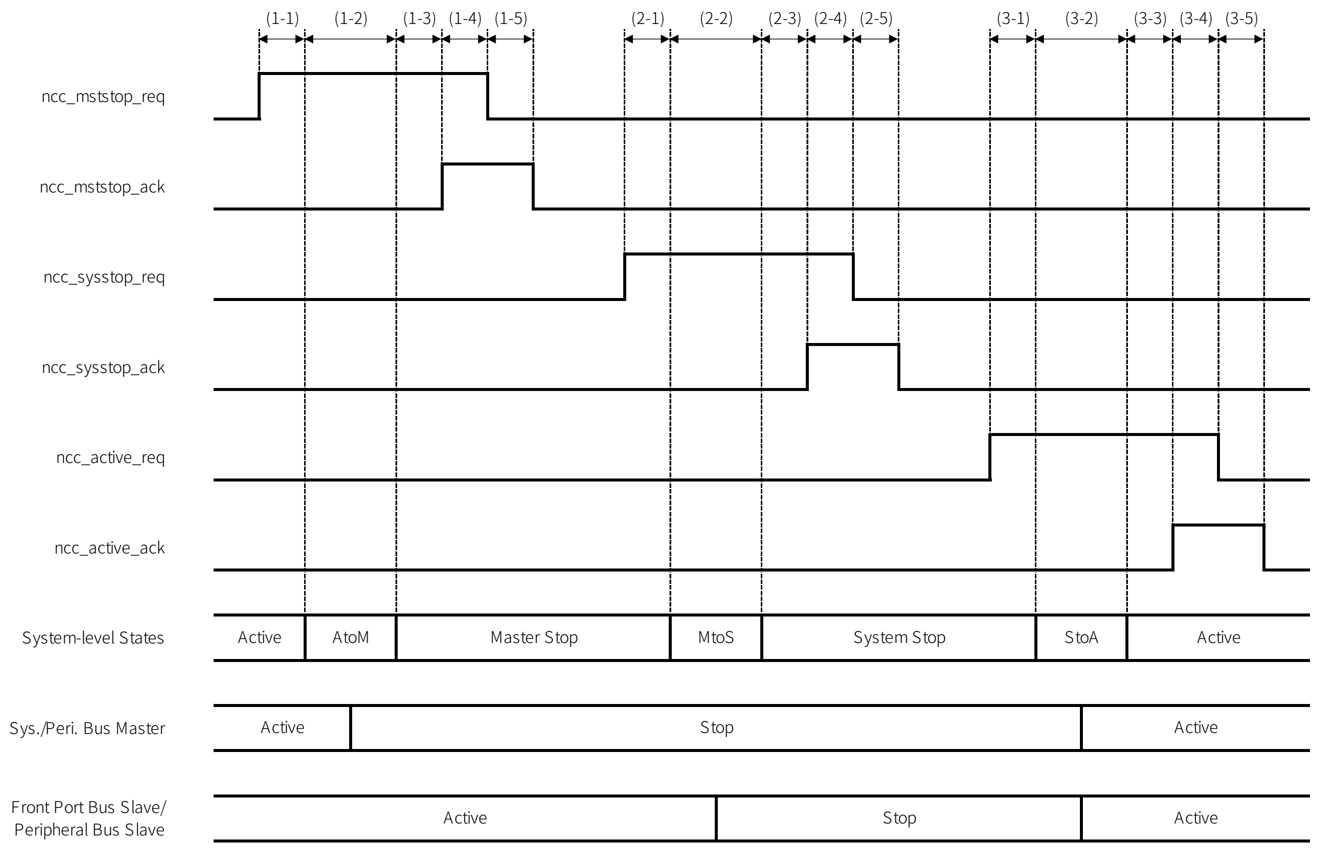


Figure 8. System-level State Transition Timing (System Stop to Active)

Table 144. System-level State Signal Timing (System Stop to Active)

Section	Interval
(1-1)	1+ clk cycle
(1-2)	Depends on the bus transaction status
(1-3)	1+ clk cycle
(1-4)	Min 1 clk cycle (external request specification)
(1-5)	1+ clk cycle
(2-1)	1+ clk cycle
(2-2)	Depends on the bus transaction status
(2-3)	1+ clk cycle
(2-4)	Min 1 clk cycle (external request specification)
(2-5)	1+ clk cycle
(3-1)	1+ clk cycle
(3-2)	1+ clk cycle
(3-3)	1+ clk cycle
(3-4)	Min 1 clk cycle (external request specification)
(3-5)	1+ clk cycle

5.3.3. System Stop to System Reset

The following shows the timing when the System Reset is performed through the System Stop state.

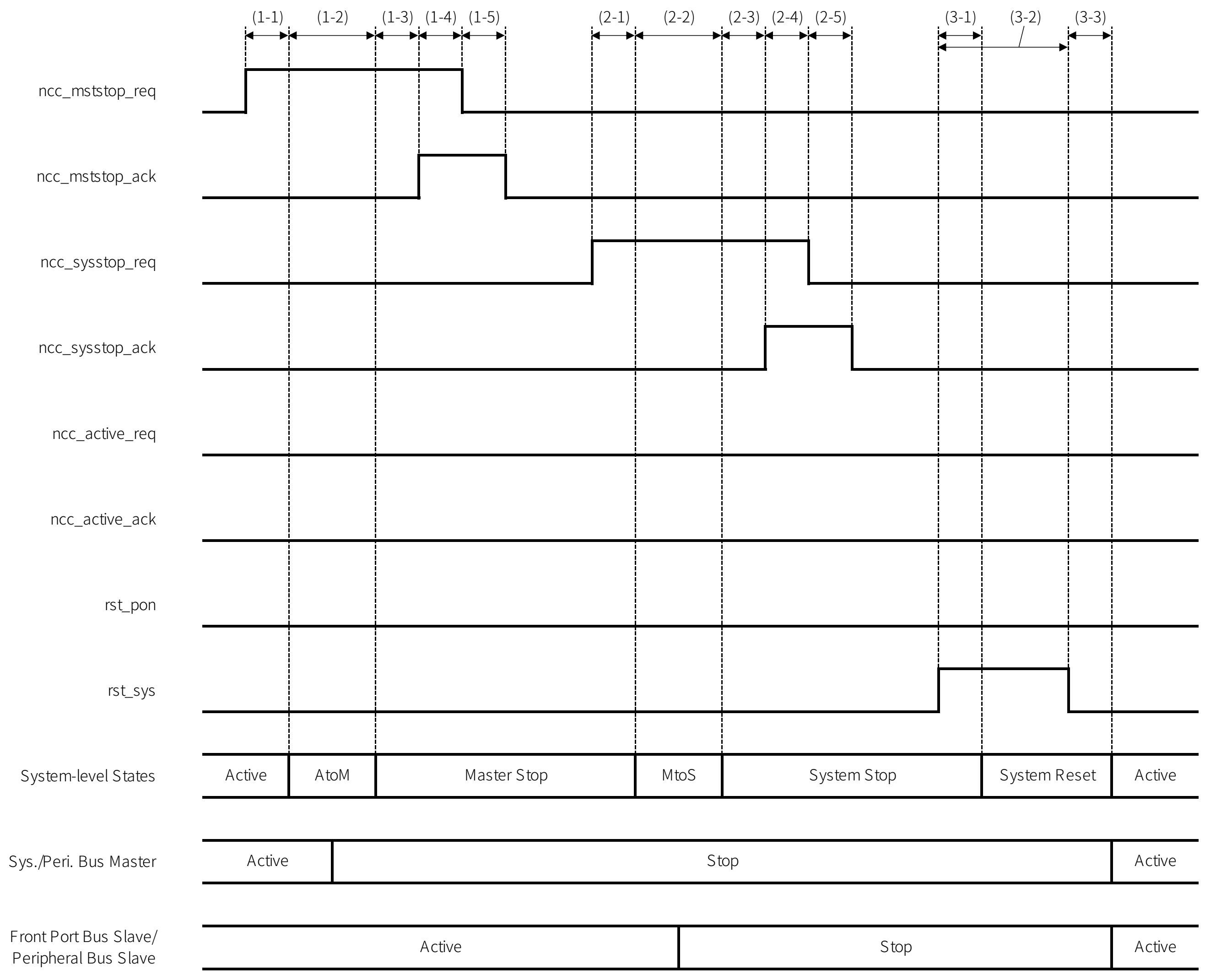


Figure 9. System-level State Transition Timing (System Stop to System Reset)

Table 145. System-level State Signal Timing (System Stop to System Reset)

Section	Interval
(1-1)	1+ clk cycle
(1-2)	Depends on the bus transaction status
(1-3)	1+ clk cycle
(1-4)	Min 1 clk cycle (external request specification)
(1-5)	1+ clk cycle
(2-1)	1+ clk cycle
(2-2)	Depends on the bus transaction status
(2-3)	1+ clk cycle
(2-4)	Min 1 clk cycle (external request specification)
(2-5)	1+ clk cycle
(3-1)	Min 1 clk cycle (external request specification)
(3-2)	Min 10 clk cycle (external request specification)
(3-3)	1+ clk cycle

"1+ clk cycle" described in Table 143, Table 144, and Table 145 indicates "the integer number of clk cycles equal to or greater than 1".

5.3.4. Debug Logic Stop

Some Debug functions and the clock supply to the DBG module can be stopped by changing the input value of the dbg_auth signal from an external pin from High to Low.
The clock supply is stopped when all the following conditions are met.

- Debug is not enabled in the dbg_auth input signal (dbg_auth = Low).
- Reset is not in progress (rst_pon=Low).

For details of the conditions for changing the dbg_auth input value and the restrictions applied when dbg_auth = Low, see NSITEXE NS31A Technical Reference Manual.

Chapter 6. Core Management Unit (CMU)

This chapter describes the Core Management Unit (CMU).

The CMU is a unit that manages the NS31A system.
It is used for controlling the reset of clocks, for configuring the system using registers as interfaces, and for providing information.

6.1. Features

6.1.1. Clock Control

The CMU performs the gating of clocks according to the request from an external pin. NS31A supports the gating of internal debug clocks using dbg_auth. For details, see NSITEXE NS31A Technical Reference Manual.

6.1.2. Reset Control

The CMU generates resets for internal use in NS31A.
After performing the retiming of the reset signals shown in Table 140, the CMU distributes the signals to each unit in NS31A. Table 146 shows the list of the reset signals that are output and distributed.

Table 146. CMU Reset Output

#	CMU Output Signal	Source Signal	Synch.	Polarity	Reset Target
1	smu_rst_dbg	rst_pon	clk	P	DBG (except DBG TAP)
2	smu_rst_ccu	rst_sys	clk	P	RVCORE
3	smu_rst_sys	rst_sys	clk	P	Other regions

For the reset specification of the entire NS31A, see Chapter 5.

6.1.3. States Control

The CMU manages System-level states, monitors Core-level states, monitors Hart-level states, and requests state transitions. For details of the individual states, see Section 3.3.

The state transition of System-level states is made by asserting ncc_mststop_req, ncc_sysstop_req, or ncc_active_req, which is input signals of NS31A. When the state transition is completed, the CMU asserts the corresponding ncc_mststop_ack, ncc_sysstop_ack, or ncc_active_ack as a completion notification.
However, the state transition is not made while Power-on Reset or System Reset is in progress, and ack is asserted immediately after req is asserted.

When ncc_mststop_req, ncc_sysstop_req, or ncc_active_req is asserted, the signal is not allowed to be negated until its corresponding ncc_mststop_ack, ncc_sysstop_ack, or ncc_active_ack is asserted.

6.1.4. Cache Clear Control

The CMU clears the internal caches in NS31A when requested from an external pin.

For information about the cache clear function, see Section 2.3.3.3.

When configuration parameter P_CACHE_SIZE_IC is 0, the cache clear function is disabled.

6.1.5. Memory-Mapped Registers

Memory-mapped registers are allocated in the Internal Register region, and the registers are used for setting the following items and for providing information.

- Hart startup control
- dbg_auth signal status
- Reset information
- Product ID
- Product revision
- Configuration parameter information

6.2. CMU Registers

6.2.1. CMU register summary

The following shows the list of registers included in CMU.
The address of each register is an offset value from the Base Address (P_ADDR_BASE_CMU).

Table 147. CMU Register List

Category	Register Name	Symbol	Address	Access	Access Protection
Status Monitor and Control	System Status Register	STATSYS	+0200H	8,16,32	-
	Core Status Register	STATCORE	+0210H	8,16,32	-
	Hart Run Status Register	STATHTRUN	+0300H	8,16,32	-
	Hart Debug Status Register	STATHTDBG	+0304H	8,16,32	-
	Hart Disable Status Register	STATHTDIS	+0308H	8,16,32	-
	Hart Run Control Register	HARTRUN	+0310H	8,16,32	-
	Hart Disable Control Register	HARTDIS	+0320H	8,16,32	-
	Cache Clear Status Register	STATCCLR	+0400H	8,16,32	-
	Cache Clear Control Register	CTRLCCLR	+0410H	8,16,32	-
Function Control	Function Control Register	FUNCCTRL	+0600H	8,16,32	-
System Communication	System Scratch Register 0	SSCRATCH0	+0700H	8,16,32	-
	System Scratch Register 1	SSCRATCH1	+0704H	8,16,32	-
Signal Monitor	Debug Signal Monitor Register	MONDBG	+0800H	8,16,32	-
	Reset Signal Monitor Register	MONRST	+0804H	8,16,32	-
	System Control Signal Monitor Register	MONSYS	+0808H	8,16,32	-
	Internal Signal Monitor Register	MONINT	+080CH	8,16,32	-
Initial Configuration	Boot Hart Configuration Register	BOOTHART	+0900H	8,16,32	-
	Initial Trap-Vector Address Configuration Register	INITMTVEC	+0904H	8,16,32	-
	Initial Reset-Vector Address Configuration Register	INITRSTVEC	+0908H	8,16,32	-
	Initial NMI-Vector Address Configuration Register	INITNMIVEC	+0910H	8,16,32	-
Information	PMP Entry Number Configuration Register	CFGPMPNUM	+0E00H	8,16,32	-
	Instruction Cache Size Configuration Register	CFGCSIZEIC	+0E04H	8,16,32	-
	Machine Architecture ID Register	MARCHID	+0FF0H	8,16,32	-
	Machine Implementation ID Register	MIMPID	+0FF4H	8,16,32	-

6.2.2. CMU register descriptions

The following describes the details of each register included in the CMU.

6.2.2.1. System Status Register (STATSYS)

STATSYS is a register that provides information about the system status of NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	STATSYS			
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 148. NCC System Status Register (STATSYS)

Bits	Field Name	Type	Reset	Description
[31:4]	-	RO	0	Reserved
[3:0]	STATSYS	RO	0	NCC System Status 4'b0000: Power-on Reset 4'b0001: System Reset 4'b0100: Active 4'b0101: A2M Transition 4'b1000: Master Stop 4'b1001: M2S Transition 4'b1010: M2A Transition 4'b1100: System Stop 4'b1101: S2A Transition

6.2.2.2. Core Status Register (STATCORE)

STATCORE is a register that indicates whether each core of the NS31A Core is active or inactive.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	STATCORE
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 149. Core Status Register (STATCORE)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	STATCORE	RO	0	Core Status 1: is in Core Active 0: is not in Core Active (in Core Inactive, Reset)

6.2.2.3. Hart Run Status Register (STATHTRUN)

STATHTRUN is a register that indicates the run status of the hart.
For information about cfg_boothart, see Section 11.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	STATHTRUN
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) Captured value of cfg_boothart on System reset release.

Table 150. Hart Run Status Register (STATHTRUN)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	STATHTRUN	RO	Captured value of cfg_boothart on System reset release	1: is in Run mode (Run or DB-Run) 0: is not in Run mode

6.2.2.4. Hart Debug Status Register (STATHTDBG)

STATHTDBG is a register that indicates whether the hart is in Debug mode or not.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	STATHTDBG
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 151. Hart Debug Status Register (STATHTDBG)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	STATHTDBG	RO	0	1: is in Debug mode (DB-Halt or DB-Run) 0: is not in Debug mode

6.2.2.5. Hart Disable Status Register (STATHTDIS)

STATHTDIS is a register that indicates whether the hart is in Disable mode or not.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	STATHTDIS
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 152. Hart Disable Status Register (STATHTDIS)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	STATHTDIS	RO	0	1: is in Disable mode 0: is not in Disable mode

6.2.2.6. Hart Run Control Register (HARTRUN)

HARTRUN is a register for changing the operation mode of the hart to the Run mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	HARTRUN
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Table 153. Hart Run Control Register (HARTRUN)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	HARTRUN	WO	0	1: Request "Run" to hart 0: No effect

6.2.2.7. Hart Disable Control Register (HARTDIS)

HARTDIS is a register for changing the hart to the Disable mode.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	HARTDIS
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO

Table 154. Hart Disable Control Register (HARTDIS)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	HARTDIS	WO	0	1: Request "Disable" to hart 0: No effect

6.2.2.8. Cache Clear Status Register (STATCCLR)

STATCCLR is a register that indicates the cache clear status and the completion of cache clearing of NS31A.
For details of cache clear control, see Section 2.3.3.2.

It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC $\geq$ 1.
When access is made in any other condition, 0 is read for read access and write access is ignored.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	COMPCCLR
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	STATCCLR	
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 155. Cache Clear Status Register (STATCCLR)

Bits	Field Name	Type	Reset	Description
[31:17]	-	RO	0	Reserved
[16]	COMPCCLR	RW	0	Cache Clear Completion Flag. This flag is set if STATCCLR=Done. Read: 1'b1: Cache clear has been completed. 1'b0: Cache clear has never been completed. Write: 1'b1 Clear COMPCCLR to 0. 1'b0 Ignored.
[15:2]	-	RO	0	Reserved
[1:0]	STATCCLR	RO	0	Cache Clear Status 2'b00: Reset/Idle 2'b01: Cache Clear Start 2'b10: Instruction Cache Clear 2'b11: Done

6.2.2.9. Cache Clear Control Register (CTRLCCLR)

CTRLCCLR is a register for requesting the cache clearing of NS31A.
When this register is read, the requested cache clear status is read.
For details of cache clear control, see Section 2.3.3.2.

It is a register that can be read/written when configuration parameter P_CACHE_SIZE_IC $\geq$ 1.
When access is made in any other condition, 0 is read for read access and write access is ignored.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	CCLRREQ
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 156. Cache Clear Control Register (CTRLCCLR)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved

Bits	Field Name	Type	Reset	Description
[0]	CCLRREQ	RW	0	Cache Clear Request Read: 0: Cache clear idle 1: Cache clear on-going (Only for a cache clear request activated by writing to CTRLCCLR) Write: 0: Ignore 1: Request Cache Clear

6.2.2.10. Function Control Register (FUNCCTRL)

FUNCCTRL is a register for controlling the functions in NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 157. Function Control Register (FUNCCTRL)

Bits	Field Name	Type	Reset	Description
[31:17]	-	RO	0	Reserved
[16]	-	RW	0	Reserved
[15:0]	-	RO	0	Reserved

6.2.2.11. System Scratch Register n (SSCRATCHn (n=0-1))

SSCRATCHn is a scratch register that can be used when communication is performed between the system and the host CPU, etc.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	SSCRATCHn															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	SSCRATCHn															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 158. System Scratch Register n (SSCRATCHn (n=0-1))

Bits	Field Name	Type	Reset	Description
[31:0]	SSCRATCHn	RW	0	Scratch register for system. Software can use this field for its own purpose.

6.2.2.12. Debug Signal Monitor Register (MONDBG)

MONDBG is a register for checking the DBG unit status by monitoring the input signal of the dbg_auth pin.
For information about the dbg_auth pin, see NSITEXE NS31A Technical Reference Manual.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBGAUTH
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) Input level of dbg_auth.

Table 159. Debug Signal Monitor Register (MONDBG)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	DBGAUTH	RO	-	Input level of dbg_auth.

6.2.2.13. Reset Signal Monitor Register (MONRST)

MONRST is a register for checking the reset status of each unit in NS31A by monitoring the input signal of each reset signal input pin.
For the list of the reset input signals and reset targets, see Table 140.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	RSTSYS	—	—	—	RSTPON
Reset:	0	0	0	0	0	0	0	0	0	0	0	(*1)	0	0	0	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) Input level of rst_XXX

Table 160. Debug Signal Monitor Register (MONRST)

Bits	Field Name	Type	Reset	Description
[31:5]	-	RO	0	Reserved
[4]	RSTSYS	RO	-	Input level of rst_sys.
[3:1]	-	RO	0	Reserved
[0]	RSTPON	RO	-	Input level of rst_pon.

The registers in NS31A can be accessed only after the System Reset is released. Therefore, 0 is always retrieved from the MONRST register.

6.2.2.14. System Control Signal Monitor Register (MONSYS)

MONSYS is a register for checking the NS31A status by monitoring the NCC System Control pin (ncc_xxx) of NS31A.
For information about NCC System Control pin, see Section 2.3.3.3 or Section 3.3.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CCLRACK	CCLRREQ
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)	(*2)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	ACTACK	ACTREQ	—	—	SSTOPACK	SSTOPREQ	—	—	MSTOPACK	MSTOPREQ
Reset:	0	0	0	0	0	0	(*1)	(*2)	0	0	(*1)	(*2)	0	0	(*1)	(*2)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) Output level of ncc_xxx
(*2) Input level of ncc_xxx

Table 161. NCC System Control Signal Monitor Register (MONSYS)

Bits	Field Name	Type	Reset	Description
[31:18]	-	RO	0	Reserved
[17]	CCLRACK	RO	-	Output level of ncc_cache_clr_ack
[16]	CCLRREQ	RO	-	Input level of ncc_cache_clr_req
[15:10]	-	RO	0	Reserved
[9]	ACTACK	RO	-	Output level of ncc_active_ack
[8]	ACTREQ	RO	-	Input level of ncc_active_req
[7:6]	-	RO	0	Reserved
[5]	SSTOPACK	RO	-	Output level of ncc_sysstop_ack
[4]	SSTOPREQ	RO	-	Input level of ncc_sysstop_req
[3:2]	-	RO	0	Reserved
[1]	MSTOPACK	RO	-	Output level of ncc_mststop_ack
[0]	MSTOPREQ	RO	-	Input level of ncc_mststop_req

6.2.2.15. Internal Signal Monitor Register (MONINT)

MONINT is a register that monitors the internal signal (smu_xxx_ack) of NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	CCLRCORE	ACTBIC	ACTDBG
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	ACTCORE	STOPBIC	STOPDBG	—	—	—	—	—	—	STOPCORE
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 162. NS31A Internal Signal Monitor Register (MONINT)

Bits	Field Name	Type	Reset	Description
[31:19]	-	RO	0	Reserved
[18]	CCLRCORE	RO	0	Signal level of Cache Clear Acknowledge from NS31A Core It can be read when configuration parameter P_CACHE_SIZE_IC $\geq$ 1. When access is made in any other condition, 0 is always read as data.
[17]	ACTBIC	RO	0	Signal level of Activation Acknowledge from bus inter-connect
[16]	ACTDBG	RO	0	Signal level of Activation Acknowledge from DBG
[15:10]	-	RO	0	Reserved
[9]	ACTCORE	RO	0	Signal level of Activation Acknowledge from NS31A Core
[8]	STOPBIC	RO	0	Signal level of System Stop Acknowledge from bus inter-connect

Bits	Field Name	Type	Reset	Description
[7]	STOPDBG	RO	0	Signal level of Master Stop Acknowledge from DBG
[6:1]	-	RO	0	Reserved
[0]	STOPCORE	RO	0	Signal level of Master Stop Acknowledge from NS31A Core

6.2.2.16. Boot Hart Configuration Register (BOOTHART)

BOOTHART is a register that reads the value of cfg_boothart, a configuration input port of NS31A, that is captured on system reset release.
For information about cfg_boothart, see Section 11.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	BOOTHART
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) Captured value of cfg_boothart on System reset release.

Table 163. Boot Hart Configuration Register (BOOTHART)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	BOOTHART	RO	Captured value of cfg_boothart on System reset release	1: hart boots after system reset release automatically. 0: hart does not boot after system reset release automatically.

6.2.2.17. Initial Trap-Vector Address Configuration Register (INITMTVEC)

INITMTVEC is a register that can read the value of cfg_mtvec, a configuration input port of NS31A, that is captured on system reset release.
For information about cfg_mtvec, see Section 11.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	INITMTVEC[31:16]															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	INITMTVEC[15:2]														—	INIT MTVEC[0]
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	0	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) Captured value of cfg_mtvec on System reset release.

Table 164. Initial Trap-Vector Address Configuration Register (INITMTVEC)

Bits	Field Name	Type	Reset	Description
[31:2]	INITMTVEC[31:2]	RO	Captured value of cfg_mtvec on System reset release.	Initial value of mtvec

Bits	Field Name	Type	Reset	Description
[1]	-	RO	0	Reserved
[0]	INITMTVEC[0]	RO	Captured value of cfg_mtvec on System reset release.	Initial value of mtvec

6.2.2.18. Initial Reset-Vector Address Configuration Register (INITRSTVEC)

INITRSTVEC is a register that can read the value of cfg_resetvec, a configuration input port of NS31A, that is captured on system reset release.
For information about cfg_resetvec, see Section 11.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	INITRSTVEC[31:16]															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	INITRSTVEC[15:2]														—	—
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) Captured value of cfg_resetvec on System reset release.

Table 165. Initial Reset-Vector Address Configuration Register (INITRSTVEC)

Bits	Field Name	Type	Reset	Description
[31:2]	INITRSTVEC[31:2]	RO	Captured value of cfg_resetvec on System reset release.	Initial value of PC
[1:0]	-	RO	0	Reserved

6.2.2.19. Initial NMI-Vector Address Configuration Register (INITNMIVEC)

INITNMIVEC is a register for changing the value of cfg_nmivec, a configuration input port of NS31A, that is captured on system reset release as the initial value.
For information about cfg_nmivec, see Section 11.1.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	INITNMIVEC[31:16]															
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	INITNMIVEC[15:2]														—	—
Reset:	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	(*1)	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO

(*1) Captured value of cfg_nmivec on System reset releas

Table 166. Initial NMI-Vector Address Configuration Register (INITNMIVEC)

Bits	Field Name	Type	Reset	Description
[31:2]	INITNMIVEC[31:2]	RW	Captured value of cfg_nmivec on System reset release	Control NMI-Vector Address.
[1:0]	-	RO	0	Reserved

6.2.2.20. PMP Entry Number Configuration Register (CFGPMPNUM)

CFGPMPNUM is a register that can read the value of P_PMPNUM, which is a configuration parameter of NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	CFGPMPNUM		
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) depend on P_PMPNUM

Table 167. PMP Entry Number Configuration Register (CFGPMPNUM)

Bits	Field Name	Type	Reset	Description
[31:3]	-	RO	0	Reserved
[2:0]	CFGPMPNUM	RO	Value set in P_PMPNUM	Number of PMP entries 3'd0: no 3'd1: 4entries 3'd2: 8entries 3'd3: not supported 3'd4: 16entries

6.2.2.21. Instruction Cache Size Configuration Register (CFGCSIZEIC)

CFGCSIZEIC is a register that can read the value of P_CACHE_SIZE_IC, which is a configuration parameter of NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	CFGCSIZEIC		
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	(*1)	(*1)	(*1)
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

(*1) depend on P_CACHE_SIZE_IC

Table 168. Instruction Cache Size Configuration Register (CFGCSIZEIC)

Bits	Field Name	Type	Reset	Description
[31:3]	-	RO	0	Reserved
[2:0]	CFGCSIZEIC	RO	Value set in P_CACHE_SIZE_IC	Instruction cache size 3'd0: no 3'd1: 4KB 3'd2: 8KB 3'd3: 16KB 3'd4: 32KB

6.2.2.22. Machine Architecture ID Register (MARCHID)

MARCHID is a register that can read the micro architecture ID of NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	MARCHID															
Reset:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	MARCHID															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 169. Machine Architecture ID Register (MARCHID)

Bits	Field Name	Type	Reset	Description
[31:0]	MARCHID	RO	80000005h	NSITEXE NS31A

6.2.2.23. Machine Implementation ID (MIMPID)

MIMPID is a register for reading the Implementation ID of NS31A.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	MIMPID															
Reset:	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	MIMPID															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 170. Machine Implementation ID (MIMPID)

Bits	Field Name	Type	Reset	Description
[31:0]	MIMPID	RO	02000004h	Revision

Chapter 7. Advanced Core Local Interruptor (ACLINT)

This chapter describes the Advanced Core Local Interruptor (ACLINT).

7.1. Features

The ACLINT has the following functions.

Table 171. ACLINT Functions

Function	Description
Timer Interrupt	The ACLINT has a 64-bit timer counter and a compare register for the Timer Interrupt. The Timer Interrupt requests an interrupt if the value of the timer counter is equal to or larger than the value of the compare register.
Software Interrupt	Request an interrupt by setting a register in the ACLINT.

The ACLINT is compliant with the RISC-V Advanced Core Local Interruptor Specification.
Note that the S-mode-related functions are not implemented.

7.2. Registers

This section describes the registers of the ACLINT.

7.2.1. Register Summary

Table 172 lists the registers of the ACLINT.
The address of each register is an offset value from the Base Address (P_ADDR_BASE_ACLINT).

Table 172. ACLINT Register Summary

Category	Register Name	Symbol	Address	Access
MSWI (Machine-level Software Interrupt Device)	Machine Software Intrrupt Pending Register	msip	+0000H	8,16,32
MTIMER (Machine-level Timer Device)	Machine Time Compare Register	mtimecmp	+4000H	8,16,32
	Machine Time Compare Register High	mtimecmph	+4004H	8,16,32
	Machine Time Control Register	mtimectrl	+D000H	8,16,32
	Machine Time Register	mtime	+BFF8H	8,16,32
	Machine Time Register High	mtimeh	+BFFCH	8,16,32
Configuration	MSWI Implementation Configuration Register	mswicfg	+FF00H	8,16,32
	MTIMER Implementation Configuration Register	mtimercfg	+FF40H	8,16,32

7.2.2. Register Description

This section describes the registers of the ACLINT in detail.

7.2.2.1. Machine Software Intrrupt Pending Register (msip)

msip is a register that controls Software Interrupt requests for each hart.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	sip
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 173. Machine Software Intrrupt Pending Register (msip)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	sip	RW	0	Software Interrupt request 1: Requested 0: Not requested

7.2.2.2. Machine Time Compare Register (mtimecmp)

mtimecmp is a register used to set the compare value for the Timer Interrupt.
Functionally, the timer compare value is 64 bits. This register can access the lower 32 bits of those bits.
The upper 32 bits can be accessed from mtimecmph.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mtimecmp[31:16]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mtimecmp[15:0]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 174. Machine Time Compare Register (mtimecmp)

Bits	Field Name	Type	Reset	Description
[31:0]	mtimecmp[31:0]	RW	0	Lower 32 bits of the timer compare value

7.2.2.3. Machine Time Compare Register High (mtimecmph)

mtimecmph is a register used to set the compare value for the Timer Interrupt.
Functionally, the compare value of the timer is 64 bits. This register can access the upper 32 bits of those bits.
The lower 32 bits can be accessed from mtimecmp.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mtimecmp[63:48]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mtimecmp[47:32]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 175. Machine Time Compare Register High (mtimecmph)

Bits	Field Name	Type	Reset	Description
[31:0]	mtimecmp[63:32]	RW	0	Upper 32 bits of the timer compare value

7.2.2.4. Machine Time Control Register (mtimectrl)

mtimectrl is a register used to control the timer for the Timer Interrupt.
This register is an NS31A unique extended register.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	haltstop
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW

Table 176. Machine Time Control Register (mtimectrl)

Bits	Field Name	Type	Reset	Description
[31:1]	-	RO	0	Reserved
[0]	haltstop	RW	1	Stop mtime count in Debug-mode 1: mtime count is stopped if hart is DB-Halt or DB-Run 0: mtime count is not stopped even if hart is DB-Halt or DB-Run

7.2.2.5. Machine Time Register (mtime)

mtime is a timer counter register for the Timer Interrupt.
Functionally, the timer counter size is 64 bits. This register can access the lower 32 bits of those bits.
The upper 32 bits can be accessed from mtimeh.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mtime[31:16]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mtime[15:0]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 177. Machine Time Register (mtime)

Bits	Field Name	Type	Reset	Description
[31:0]	mtime[31:0]	RW	0	Lower 32 bits of the timer counter

7.2.2.6. Machine Time Register High (mtimeh)

mtimeh is a timer counter register for the Timer Interrupt.
Functionally, the timer counter size is 64 bits. This register can access the upper 32 bits of those bits.
The lower 32 bits can be accessed from mtime.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mtime[63:48]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mtime[47:32]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 178. Machine Time Register High (mtimeh)

Bits	Field Name	Type	Reset	Description
[31:0]	mtime[63:32]	RW	0	Upper 32 bits of the timer counter

7.2.2.7. MSWI Implementation Configuration Register (mswicfg)

mswicfg is a register that indicates the Software Interrupt implementation configuration.
This register is an NS31A unique extended register.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mtimercfg															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mtimercfg															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 179. MSWI Implementation Configuration Register (mswicfg)

Bits	Field Name	Type	Reset	Description
[31:0]	mswicfg	RO	00000001H	Software Interrupt implementation configuration Always 1 for NS31A.

7.2.2.8. MTIMER Implementation Configuration Register (mtimercfg)

mtimercfg is a register that indicates the Timer Interrupt implementation configuration.
This register is an NS31A unique extended register.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	mswicfg															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	mswicfg															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 180. MTIMER Implementation Configuration Register (mtimercfg)

Bits	Field Name	Type	Reset	Description
[31:0]	mtimercfg	RO	00000001H	Timer Interrupt implementation configuration Always 1 for NS31A.

7.3. Operations

7.3.1. Timer Interrupt

After the reset is released, mtime starts counting up from 0. mtime stops the count-up in the following cases.

- The input port ncc_mtime_cnten is 0.
- mtimectrl.haltstop is set to 1, and the Hart-level states is DB-Halt or DB-Run.

If the value of mtime is equal to or larger than the value of mtimecmp, the Timer Interrupt is requested. If the value of mtime is smaller than the value of mtimecmp, the Timer Interrupt is not requested.

mtime and mtimecmp are cleared to zero by reset. According to this operation, the ACLINT generates a Timer Interrupt request immediately after the reset. Set the timer value of mtimecmp to prevent a Timer Interrupt request from being generated in advance of enabling the Timer Interrupt.

7.3.2. Software interrupt

If 1 is written in msip.sip, the Software Interrupt is requested. If 0 is written in msip.sip, the Software Interrupt request is canceled.

Chapter 8. Platform-Level Interrupt Controller (PLIC)

This chapter describes the Platform-Level Interrupt Controller (PLIC).

8.1. Features

The PLIC processes the External Interrupt factor and notifies the hart of the External Interrupt request. The functional details are as follows.

- The interrupt input requested from the outside is processed according to the processing rules, and an interrupt request is issued to the hart.
- Priority can be set for each interrupt.
- Among the interrupts exceeding the preset threshold for priority, the interrupt with the highest priority is requested.
- If multiple effective interrupts have the same priority, the interrupt with a smaller interrupt SourceID (bit position number of the int_extint input) takes the priority.
- The detection method, level or edge, can be defined for each factor using the configuration parameter P_EXTINT_EDGE.
- The number of interrupt input signals from the outside can be set using the configuration parameter P_EXTINT_WIDTH.

The PLIC is compliant with the RISC-V Platform-Level Interrupt Controller Specification.

8.2. Registers

This section describes the registers of the PLIC.

8.2.1. Register Summary

Table 181 lists the registers of the PLIC.
The address of each register is an offset value from the Base Address (P_ADDR_BASE_PLIC).

Table 181. PLIC Register Summary

Category	Register name	Symbol	Address	Access
External Interrupt	External Interrupt Source Priority Register (N=1-255)	eispriN	+000000H +Nx4H	8,16,32
	External Interrupt Source Pending Register (N=0-7)	eispenN	+001000H +Nx4H	8,16,32
	External Interrupt Hart Enable Register (N=0-7)	eihenN	+002000H +Nx4H	8,16,32
	External Interrupt Source Flag Status Register (N=0-7)	eisfstatN	+1F8000H +Nx4H	8,16,32
	External Interrupt Source Flag Clear Register (N=0-7)	eisfclrN	+1F9000H +Nx4H	8,16,32
	External Interrupt Source Type Configuration Register (N=0-7)	eicfgstN	+1FC000H +Nx4H	8,16,32
	External Interrupt Hart Implementation Configuration Register	eicfghi	+1FD000H	8,16,32
	External Interrupt Hart Threshold Register	eihth	+200000H	8,16,32
	External Interrupt Hart Claim/complete ID Register	eihcid	+200004H	8,16,32

8.2.2. Register Description

This section describes the registers of the PLIC in detail.

8.2.2.1. External Interrupt Source Priority Register (eispriN, N=1-255)

eispriN is a register used to set the priority order of External Interrupts.
The interrupt priority is variable, and the interrupt with the larger number has the higher priority. If multiple interrupts have the same priority, the interrupt with a smaller interrupt SourceID (bit position number of the int_extint input) takes the priority.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	—	—	—	—	—	—	—	—	eispri							
Reset:	0	0	0	0	0	0	0	0	*1	*1	*1	*1	*1	*1	*1	*1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	*1	*1	*1	*1	*1	*1	*1	*1

*1 Depend on P_EXTINT_WIDTH and P_EXTINT_PRIWIDTH

Table 182. External Interrupt Source Priority Register (eispriN, N=1-255)

Bits	Field Name	Type	Reset	Description
[31:8]	-	RO	0	Reserved
[7:0]	eispri	*1	*1	Interrupt priority. Larger eispri has higher priority. If all eispri bits are 0, the external interrupt from int_extint[N] is disabled. *1 Depend on P_EXTINT_WIDTH and P_EXTINT_PRIWIDTH When P_EXTINT_WIDTH≥N - [7:P_EXTINT_PRIWIDTH] RO, 0 - [(P_EXTINT_PRIWIDTH-1):0] RW, 1 When P_EXTINT_WIDTH<N - [7:0] RO, 0

8.2.2.2. External Interrupt Source Pending Register (eispenN, N=0-7)

eispenN is a register that indicates the Pending status of an External Interrupt.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	eispen															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	eispen															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 183. External Interrupt Source Pending Register (eispenN, N=0-7)

Bits	Field Name	Type	Reset	Description
[31:0]	eispen	RO	0	[31] int_extint[31+32*N] : [0] int_extint[0+32*N] 1:Pending 0:No Pending The bit corresponding to an unused input int_extint[p] (p=0, p>P_EXTINT_WIDTH) is RO, which is set to 0.

8.2.2.3. External Interrupt Hart Enable Register (eihenN, N=0-7)

eihenN is a register used to set whether the PLIC acknowledges External Interrupts.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	eihen															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	eihen															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 184. External Interrupt Hart Enable Register (eihenN, N=0-7)

Bits	Field Name	Type	Reset	Description
[31:0]	eihen	RW	0	[31] int_extint[31 + 32*N] : [0] int_extint[0 + 32*N] 1: Enable interrupt 0: Disable interrupt The bit corresponding to an unused input int_extint[p] (p=0, p>P_EXTINT_WIDTH) is RO, which is set to 0.

8.2.2.4. External Interrupt Source Flag Status Register (eisfstatN,N=0-7)

eisfclrN is a register used to clear the pending interrupt flag of the External Interrupt Source Input. This is an NS31A unique register.
For information about the pending interrupt flag, see Section 8.3.2.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	eisfstat															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	eisfstat															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 185. External Interrupt Source Flag Status Register (eisfstatN, N=0-7)

Bits	Field Name	Type	Reset	Description
[31:0]	eisfstat	RO	0	[31] int_extint[31 + 32*N] : [0] int_extint[0 + 32*N] 1: The pending interrupt flag state is 1 0: The pending interrupt flag state is 0 The bit corresponding to an unused input int_extint[p] (p=0, p>P_EXTINT_WIDTH) is RO, which is set to 0.

8.2.2.5. External Interrupt Source Flag Clear Register (eisfclrN, N=0-7)

eisfclrN is a register used to clear the pending interrupt flag of the External Interrupt Source Input. This is an NS31A unique register.
For information about the pending interrupt flag, see Section 8.3.2.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	eisfclr															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	eisfclr															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Table 186. External Interrupt Source Flag Clear Register (eisfclrN, N=0-7)

Bits	Field Name	Type	Reset	Description
[31:0]	eisfclr	WO	0	[31] int_extint[31 + 32*N] : [0] int_extint[0 + 32*N] 1: Clear pending interrupt flag 0: Do nothing The bit corresponding to an unused input int_extint[p] (p=0, p>P_EXTINT_WIDTH) is RO, which is set to 0.

8.2.2.6. External Interrupt Source Type Configuration Register (eicfgstN, N=0-7)

eicfgstN is a register that reads the External Interrupt Input detection method set in the configuration parameter P_EXTINT_EDGE. This is an NS31A unique register.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	eicfgst															
Reset:	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	eicfgst															
Reset:	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

*1 P_EXTINT_EDGE[31+32*N:32*N]

Table 187. External Interrupt Source Type Configuration Register (eicfgstN, N=0-7)

Bits	Field Name	Type	Reset	Description
[31:0]	eicfgst	RO	P_EXTINT_EDGE[31+32*N:32*N]	[31] int_extint[31 + 32*N] : [0] int_extint[0 + 32*N] 1: Edge 0: Level The bit corresponding to an unused input int_extint[p] (p=0, p>P_EXTINT_WIDTH) is RO, which is set to 0.

8.2.2.7. External Interrupt Hart Implementation Configuration Register (eicfghi)

eicfghi is a register that reads the External Interrupt implementation configuration. This is an NS31A unique register.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	eicfghi															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	eicfghi															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 188. External Interrupt Hart Implementation Configuration Register (eicfghi)

Bits	Field Name	Type	Reset	Description
[31:0]	eicfghi	RO	00000001H	External Interrupt implementation configuration Always 1 for NS31A.

8.2.2.8. External Interrupt Hart Threshold Register (eihth)

eihth is a register used to set the threshold for the priority of the External Interrupts received by the PLIC. Interrupts with higher priority than eihth are valid. If you do not use the eihth function, set 0.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	eihth							
Reset:	0	0	0	0	0	0	0	0	*1	*1	*1	*1	*1	*1	*1	*1
Type:	RO	RO	RO	RO	RO	RO	RO	RO	*1	*1	*1	*1	*1	*1	*1	*1

*1 Depend on P_EXTINT_PRIWIDTH

Table 189. External Interrupt Hart Threshold Register (eihth)

Bits	Field Name	Type	Reset	Description
[31:8]	-	RO	0	Reserved
[7:0]	eihth	*1	*1	Threshold of interrupt priority. If all eihth bits are 1, the external interrupt is disabled for the hart. *1 Depend on P_EXTINT_PRIWIDTH - [7:P_EXTINT_PRIWIDTH] RO, 0 - [P_EXTINT_PRIWIDTH-1:0] RW, 1

8.2.2.9. External Interrupt Hart Claim/complete ID Register (eihcid)

eihcid is a register that indicates the interrupt SourceID corresponding to the External Interrupt request source. It is also used for the hart to notify PLIC of the completion of interrupt processing. For details, see Section 8.3.2.

Bits:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field Name:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field Name:	–	–	–	–	–	–	–	–	eihcid							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type:	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW

Table 190. External Interrupt Hart Claim/complete ID Register (eihcid)

Bits	Field Name	Type	Reset	Description
[31:8]	-	RO	0	Reserved
[7:0]	eihcid	RW	0	ID of the highest priority External Interrupt. 0 means no pending interrupt.

8.3. Operations

The PLIC detects an external interrupt request from the int_extint port and outputs it to the hart.
For the detection method for input interrupt signals, edge or level can be selected using the configuration parameter (P_EXTINT_EDGE).
This section describes the software settings and shows an example of the interrupt processing flow.

8.3.1. Software Settings

1. Set the priority in the eispri register corresponding to the int_extint to be used. The higher the number, the higher the priority. If multiple interrupts have the same priority, the one with the smaller SourceID takes the priority.
2. Set the eihenN and eihth registers corresponding.

8.3.2. Example of the Interrupt Processing Flow

1. The input interrupt signal int_extint from each Source is asserted.
2. When the PLIC detects the assertion of int_exting (in step 1), it sets the corresponding bit of the eispenN register. The setting condition differs depending on the detection method; level or edge.
    - In the case of level detection:
      The assertion is ignored temporarily if the previous interrupt is pending. The interrupt request is issued if the signal remains asserted after the previous interrupt processing completes.
    - In the case of edge detection:
      If the trigger is detected and the previous interrupt is pending, the pending interrupt flag inside the PLIC is set to 1 and the interrupt request is issued again after the previous interrupt processing completes. After the interrupt request is issued, the pending interrupt flag is cleared to 0.
3. The PLIC chooses the valid interrupt requests according to the eithenN register setting.
4. If there are multiple valid interrupt requests, the PLIC selects the interrupt request with the highest priority, using the priority of each interrupt that is set in eispriN and the SourceID. It sets the ID number of this request in the eihcid register.
5. If the priority eispriN of the interrupt request selected in step 4 is higher than the threshold eihth, the PLIC sends the interrupt request to the hart.
6. Upon detecting the interrupt request, the hart starts executing the handler for the External Interrupt. Then it obtains the SourceID that issued the interrupt request (Claim Response) by reading the Claim/complete ID register eihcid in the PLIC (Interrupt Claim).
7. Upon detecting a Claim Response, the PLIC clears the corresponding bit of the Pending register eispenN of the corresponding interrupt factor Source.
8. If the detection method for the Source input interrupt request is level detection, the hart clears the interrupt factor Source to clear the interrupt request to the PLIC.
9. After completing the interrupt processing, the hart writes the SourceID obtained in step 6 to the Claim/complete ID register eihcid (Interrupt Completion).
10. The PLIC terminates the interrupt input of the written SourceID and resumes the interrupt enable state. At this time, the PLIC does not check whether the written value matches the ID read in the Claim Response.
11. If there is any interrupt request remaining, the processing is repeated from step 4.

Chapter 9. Debug Subsystem (DBG)

This chapter describes the Debug Subsystem (DBG).

9.1. Features

DBG provides a mechanism for debugging the NS31A programs.
The following functions are provided.

Table 191. Functions Provided by DBG

Function	Description
JTAG interface	Access to DBG via the JTAG interface
Debug Execution Control	Controls the Halt/Resume of the hart

Table 192 shows the DBG component modules for implementing the functions listed in Table 191.

Table 192. DBG Component Modules

Module name	Description
Debug Module (DM)	Execution control module of the hart
Debug Transfer Module (DTM)	Interface module for DM access (JTAG TAP)

Figure 10 shows the block diagram of NS31A DBG.

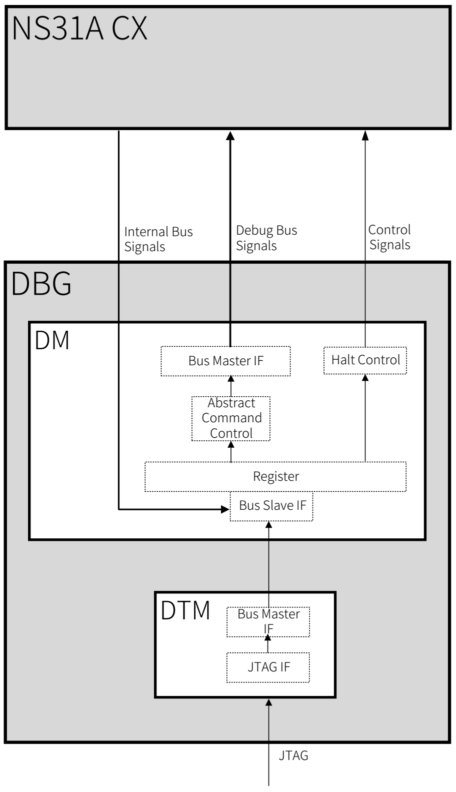


Figure 10. DBG Block Diagram

For details of the Debug subsystem, see NSITEXE NS31A Technical Reference Manual.

Chapter 10. Functional Safety

This chapter describes the hardware functional safety mechanisms of NS31A.
For details of the recommended settings and use examples of related registers, see NSITEXE NS31A Hardware Safety Manual.

10.1. Overview

NS31A is equipped with the following hardware safety mechanisms.

Memory Protection

When configuration parameter P_PMPNUM ≥ 1, Physical Memory Protection (PMP) detects illegal access from bus master to memory or peripheral circuit in order to protect memories and peripheral circuits.
When P_PMPNUM is 0, PMP is not provided.

Bus Error Detection

In the AXI configuration, the internal bus has the following safety mechanisms.

- Bus error response monitor
- Bus protocol checker

This does not apply to the AHB configuration, which does not have any internal bus.

Hardware Redundancy (Optional)

When configuration parameter P_REDUNDANT is set to 2, the main control and arithmetic circuits of NS31A have a Dual Core Lockstep (DCLS) configuration.

RAM ECC (Optional)

When configuration parameter P_REDUNDANT is set to 1 or 2, ECC-based error detection and correction is performed on the cache in NS31A and the ILM and DLM RAMs.
RAM ECC is disabled when the cache, ILM, and DLM are not mounted based on the configuration parameter settings.

Bus Interface Protection (Optional)

When configuration parameter P_REDUNDANT is set to 2, the external bus interface of NS31A has the following safety mechanisms in the AXI configuration.

- ECC protection of payload signals
- Parity protection of handshake signals

This does not apply to the AHB configuration. Implement appropriate safety mechanisms using the external AHB bus.

Chapter 11. Interface Usage Notes

This chapter describes the notes regarding the NS31A interface.
There are the following items.

- Setting of reset initial value signals
- DFT signal control

11.1. Setting of Reset Initial Value Signals

It is possible to set the operation mode of the processor and the initial values of registers used when the reset is released, by using the pin signals shown in Table 193.

Table 193. Config Port List

Signal	Width	Supported Range	Description
cfg_boothart	1	Any	Set the operation mode after the reset of the hart is released. The setting values of the bits are as follows. 0: The operation mode after the reset release is idle. 1: The operation mode after the reset release is run. When the operation mode after the reset release is idle, the hart can be put in the run mode by setting the HARTRUN register or ncc_runhart pin to 1. For details of the operation mode, see Section 3.3.3.
cfg_mtvec	32	[31:2] Any [1] 0 [0] 0 or 1	Set the initial value of mtvec of the hart. For details of mtvec, see RISC-V Priv. Arch. Spec. [31:2] Specify the initial value used after releasing the reset of the BASE bit. * If the MODE bit is 1 (Vectored), [6:2] are ignored and, therefore, it needs to be aligned to 128 bytes. [1] Fixed to 0. [0] Specify the initial value used after releasing the reset of the MODE bit.
cfg_resetvec	32	[31:2] Any [1:0] 0	Set the initial value of mhtpc of the hart. For details of mhtpc, see Section 3.8.3.2.3. If the hart directly enters the run mode after the reset is released, this value is applied to the default value of the PC. [31:2] Specify the Reset Vector address. [1:0] Fixed to 0.
cfg_nmivec	32	[31:2] Any [1:0] 0	Set the initial value of NMI Vector Address in each Core. For information about the NMI, see Section 3.4. [31:2] Specify the NMI Vector address. [1:0] Fixed to 0.

For these signals, capturing of the initial value starts during the System Reset, and then finishes within 5 clks after the System Reset is released to finalize the initial value.
Be sure to finalize the initial value for these signals before the System Reset is released, and retain the finalized value until 5 clks have elapsed after the System Reset is released.

11.2. DFT Signal Control

To support DFT in IP, NS31A has the pins shown in Table 194. Under normal use, NS31A requires that the level of the signals be fixed appropriately according to the instructions given in Table 194.

Table 194. DFT Port List

Signal	Supported Value	Description
rst_dft	0: Not reset 1: Reset	This is the reset signal to be used for DFT. While the signal is not used during the normal operation, it is recommended to fix it to 1.
dft_mask_reset	0: Not Mask 1: Mask	This signal is used to mask the reset signal from the outside. Fix it to 0 during the normal operation.
dft_bypass_clk_inv	0: Not Bypass 1: Bypass	This signal is used to bypass the internal clock inverter. Fix it to 0 during the normal operation.
dft_bypass_int_rst	0: Not Bypass 1: Bypass	This signal is used to bypass the internal reset generation circuit. Fix it to 0 during the normal operation.
dft_disable_gck	0: Enable clock gating 1: Disable clock gating	This signal is used to disable the internal clock gating function. Fix it to 0 during the normal operation.

Chapter 12. NS31A Configuration Parameter and Port List

12.1. Configuration Parameter List

Table 195 lists the configuration parameters of NS31A.

Table 195. NS31A Configuration Parameter List

Parameter	AXI Config. Default Value	AHB Config. Default Value	Supported Range	Description
P_RV32E	0	1	0-1	Specify the Base ISA. 0:RV32I 1:RV32E To enable this function, P_FPU must be set to 0 (disable the RV32F instruction). [*1]
P_RV32C	0	1	0-1	Specify whether to support RV32C instructions. 0: Disable 1: Enable
P_RV32B	1	1	0-1	Specify whether to support RV32B instructions. 0: Disable 1: Enable
P_FPU	1	0	0-1	Specify whether to support RV32F instructions. 0: Disable 1: Enable To enable this mechanism, P_RV32E must be set to 0. [*1]
P_FPU_EXP2LOG2	0	0	0-1	Specify whether to support NS31A's unique additional instructions (ns.fexp2.s and ns.flog2.s). 0: Disable 1: Enable To enable this function, P_FPU must be set to 1 (enable the RV32F instruction). [*2]
P_ADDR_BASE_ILM	0180_0000_H	-	Any 32bit value	Specify the ILM Region base address. (The region from the specified value to the value specified by P_ADDR_SIZE_ILM is allocated.) The specified value needs to be aligned to the size specified in P_ADDR_SIZE_ILM.[*3]
P_ADDR_BASE_DLM	7000_0000_H	-	Any 32bit value	Specify the DLM Region base address. (The region from the specified value to the value specified by P_ADDR_SIZE_DLM is allocated.) The specified value needs to be aligned to the size specified in P_ADDR_SIZE_DLM. [*3]
P_ADDR_BASE_DBG	0000_0000_H	0000_0000_H	Any 32bit value	Specify the Debug Subsystem Register Region base address. (A 16-KB region from the specified value is allocated.) The specified value needs to be aligned to 16 KB. [*3]
P_ADDR_BASE_HCR	0000_8000_H	0000_8000_H	Any 32bit value	Specify the Hart Context Register Region base address. (A 32-KB region from the specified value is allocated.) The specified value needs to be aligned to 32 KB.[*3]
P_ADDR_BASE_ACLIN T	0200_0000_H	0200_0000_H	Any 32bit value	ACLINT Register Region base address. (A 64-KB region from the specified value is allocated.) The specified value needs to be aligned to 64 KB.[*3]
P_ADDR_BASE_CMU	0210_0000_H	0210_0000_H	Any 32bit value	CMU Register Region base address. (A 4-KB region from the specified value is allocated.) The specified value needs to be aligned to 4 KB.[*3]
P_ADDR_BASE_REG	0210_1000_H	0210_1000_H	Any 32bit value	Specify the Internal Register Region base address. (A 4-KB region from the specified value is allocated.) The specified value needs to be aligned to 4 KB.[*3]

Parameter	AXI Config. Default Value	AHB Config. Default Value	Supported Range	Description
P_ADDR_BASE_PLIC	0C00_0000$_H$	0C00_0000$_H$	Any 32bit value	PLIC Register Region base address. (A 4-MB region from the specified value is allocated.) The specified value needs to be aligned to 4 MB.[*3]
P_ADDR_BASE_PER	2000_0000$_H$	-	Any 32bit value	Specify the External Peripheral Bus Region base address. (The region from the specified value to the value specified by P_ADDR_SIZE_PER is allocated.) The specified value needs to be aligned to the size specified in P_ADDR_SIZE_PER.[*3]
P_ADDR_SIZE_ILM	3	-	0-12	Specify the ILM Region size. 0: None 1: 4 KB 2: 8 KB 3: 16 KB 4: 32 KB 5: 64 KB - 12: 8 MB
P_ADDR_SIZE_DLM	5	-	1-12	Specify the DLM Region size. 1: 4 KB 2: 8 KB 3: 16 KB 4: 32 KB 5: 64 KB - 12: 8 MB
P_ADDR_SIZE_PER	18	-	0-18	Specify the External Peripheral Bus Region size. 0: None 1: 4 KB 2: 8 KB - 18: 512 MB
P_CACHE_SIZE_IC	3	0	0-4	Specify the instruction cache size. 0: None 1: 4 KB 2: 8 KB 3: 16 KB 4: 32 KB
P_PMPNUM	4	0	0-2,4	Specify the number of PMP entries. 0: None 1: 4 entries 2: 8 entries 3: Reserved 4: 16 entries
P_BTB	1	0	0-1	Branch prediction mechanism with the BTB 0: Disable 1: Enable
P_M_AHB	0	1	0-1	Select NS31A configuration. 0: AXI configuration 1: AHB configuration
P_BUS_PROT0_PRIV	0	0	0-1	Select the Bus Access Privileged configuration. 0: Always Privileged access 1: Privileged access applied to the NS31A Core when in M-Mode, to the Front Port Bus Slave when AWPROT[0]/ARPROT[0] is 1, and to the Peripheral Bus Slave when PPROT[0] is 1.
P_FBS_AXID_WIDTH	16	-	4-16	Front Port Bus Slave AxID bit width
P_SBM_AXPROT1_FVAL	1	-	0-1	ARPROT[1] and AWPROT[1] output by the System Bus Master
P_IBM_HPROT3_2_FVAL	-	1	0-3	HPROT[3:2] output by the Instruction Bus Master HPROT[3] Modifiable HPROT[2] Bufferable
P_DBM_HPROT3_2_FVAL	-	1	0-3	HPROT[3:2] output by the Data Bus Master HPROT[3] Modifiable HPROT[2] Bufferable
P_SBM_AXCACHE_FVAL	0	-	4'b0000-4'b1111	ARCACHE[3:0] and AWCACHE[3:0] output by the System Bus Master.
P_EXTINT_WIDTH	8	8	1-255	Number of External Interrupts input from the outside of NS31A.
P_EXTINT_PRIWIDTH	4	4	1-7	External Interrupt priority bit width from the outside of NS31A.

Parameter	AXI Config. Default Value	AHB Config. Default Value	Supported Range	Description
P_EXTINT_EDGE	0	0	Any P_EXTINT_WIDTH+1 bits value	External Interrupt input edge or level setting from the outside of NS31A Bit[P_EXTINT_WIDTH] int_extint[P_EXTINT_WIDTH] : Bit[1] int_extint[1] Bit[0] Reserved 1： Edge 0： Level
P_HARTID	0000_0000$_H$	0000_0000$_H$	0-31	Hart ID NS31A supports 0 to 31 as the hart IDs.
P_SYNC_STAGES_CLK	2	2	2-5	Number of FF stages in the CDC synchronous circuit (JTAG clock to System clock)
P_SYNC_STAGES_TCK	2	2	2-5	Number of FF stages in the CDC synchronous circuit (System clock to JTAG clock)
P_REDUNDANT	0	0	0-2	Functional safety mechanism (other than the NS31A CX layer) 0: None 1: RAM ECC 2: DCLS, RAM ECC, and BUS EDC
P_MUL_SMALL	0	1	0-1	Specify the multiplier for RV32M instructions. 0:Fast multiplier (high-performance) 1:Small multiplier (area-efficient)
P_DIV_SMALL	0	1	0-1	Specify the divider for RV32M instructions. 0:Fast divider (high-performance) 1: Small divider (area-efficient)
P_HPMNUM	1	0	0-8	Specify the number of HPM counters 0: None 1-8: Number of HPM counters

1. When RV32E is selected for the Base ISA, do not enable the RV32F instruction. It is not supported by the compiler.
2. When enabling the ns.fexp2.s or ns.flog2.s instruction, be sure to enable the RV32F Instruction.
3. Make sure that the address regions specified by these configuration parameters do not overlap.

12.2. Port List

This section lists the pins of the individual NS31A layers in the AXI configuration.

- NS31A UIL
- NS31A CX
- NS31A DBG
- NS31A MEM

The pins of the individual NS31A layers in the AHB configuration are also listed.

- NS31A UIL
- NS31A CX
- NS31A DBG
- NS31A MEM

The NS31A UIL and MEM layers can be changed by the user. Therefore, if any change is made, the information shown in these tables may not apply.
Unless otherwise noted, pins, except clocks, are synchronized with the clk clock.

12.2.1. NS31A UIL Port List (AXI Configuration)

The following table lists the pins in the UIL layer in the NS31A AXI configuration.

Table 196. NS31A UIL Port List (AXI Configuration)

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
1	Clock	clk	I	-	-	System Clock
		clk_sf	I	-	-	System Clock for redundant logic
2	Reset	rst_pon	I	clk	P	Power-on Reset
		rst_sys	I	clk	P	System Reset

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
3	System Bus Master	sbm_awid[4:0]	O	clk	P	
		sbm_awaddr[31:0]	O	clk	P	
		sbm_awlen[7:0]	O	clk	P	
		sbm_awsize[2:0]	O	clk	P	
		sbm_awburst[1:0]	O	clk	P	
		sbm_awlock	O	clk	P	
		sbm_awcache[3:0]	O	clk	P	
		sbm_awprot[2:0]	O	clk	P	
		sbm_awvalid	O	clk	P	
		sbm_awready	I	clk	P	
		sbm_awuser	O	clk	P	1: Debug, 0: Non-Debug
		sbm_wdata[31:0]	O	clk	P	
		sbm_wstrb[3:0]	O	clk	P	
		sbm_wlast	O	clk	P	
		sbm_wvalid	O	clk	P	
		sbm_wready	I	clk	P	
		sbm_bid[4:0]	I	clk	P	
		sbm_bresp[1:0]	I	clk	P	
		sbm_bvalid	I	clk	P	
		sbm_bready	O	clk	P	
		sbm_arid[4:0]	O	clk	P	
		sbm_araddr[31:0]	O	clk	P	
		sbm_arlen[7:0]	O	clk	P	
		sbm_arsize[2:0]	O	clk	P	
		sbm_arburst[1:0]	O	clk	P	
		sbm_arlock	O	clk	P	
		sbm_arcache[3:0]	O	clk	P	
		sbm_arprot[2:0]	O	clk	P	
		sbm_arvalid	O	clk	P	
		sbm_arready	I	clk	P	
		sbm_aruser	O	clk	P	1: Debug, 0: Non-Debug
		sbm_rid[4:0]	I	clk	P	
		sbm_rdata[31:0]	I	clk	P	
		sbm_rresp[1:0]	I	clk	P	
		sbm_rlast	I	clk	P	
		sbm_rvalid	I	clk	P	
		sbm_rready	O	clk	P	

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
3	System Bus Master (Cont.)	sbm_awcpayload[13:0]	O	clk	P	EDC protection code for write address channel payload [13:7] Code for sbm_awid[4:0], sbm_awlen[7:0], sbm_awsize[2:0], sbm_awburst[1:0], sbm_awlock, sbm_awcache[3:0], sbm_awprot[2:0], sbm_awuser [6:0] Code for sbm_awaddr[31:0]
		sbm_awcvalid	O	clk	P	EDC protection code for sbm_awvalid
		sbm_awcready	I	clk	P	EDC protection code for sbm_awready
		sbm_wcpayload[13:0]	O	clk	P	EDC protection code for write data channel payload [13:7] Code for sbm_wlast, sbm_wstrb[3:0] [6:0] Code for sbm_wdata[31:0]
		sbm_wcvalid	O	clk	P	EDC protection code for sbm_wvalid
		sbm_wcready	I	clk	P	EDC protection code for sbm_wready
		sbm_bcpayload[6:0]	I	clk	P	EDC protection code for write response channel payload [6:0] Code for sbm_bid[4:0], sbm_bresp[1:0]
		sbm_bcvalid	I	clk	P	EDC protection code for sbm_bvalid
		sbm_bcready	O	clk	P	EDC protection code for sbm_bready
		sbm_arcpayload[13:0]	O	clk	P	EDC protection code for read address channel payload [13:7] Code for sbm_arid[4:0], sbm_arlen[7:0], sbm_arsize[2:0], sbm_arburst[1:0], sbm_arlock, sbm_arcache[3:0], sbm_arprot[2:0], sbm_aruser [6:0] Code for sbm_araddr[31:0]
		sbm_arcvalid	O	clk	P	EDC protection code for sbm_arvalid
		sbm_arcready	I	clk	P	EDC protection code for sbm_arready
		sbm_rcpayload[13:0]	I	clk	P	EDC protection code for read data channel payload [13:7] Code for sbm_rid[4:0], sbm_rresp[1:0], sbm_rlast [6:0] Code for sbm_rdata[31:0]
		sbm_rcvalid	I	clk	P	EDC protection code for sbm_rvalid
		sbm_rcready	O	clk	P	EDC protection code for sbm_rready

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
4	Front Port Bus Slave	fbs_awid[P_FBS_AXID_WIDTH-1:0]	I	clk	P	The port width is specified by a parameter.
		fbs_awaddr[31:0]	I	clk	P	
		fbs_awlen[7:0]	I	clk	P	
		fbs_awsize[2:0]	I	clk	P	
		fbs_awburst[1:0]	I	clk	P	
		fbs_awlock	I	clk	P	
		fbs_awcache[3:0]	I	clk	P	
		fbs_awprot[2:0]	I	clk	P	
		fbs_awvalid	I	clk	P	
		fbs_awready	O	clk	P	
		fbs_awuser	I	clk	P	1: Debug, 0: Non-Debug
		fbs_wdata[31:0]	I	clk	P	
		fbs_wstrb[3:0]	I	clk	P	
		fbs_wlast	I	clk	P	
		fbs_wvalid	I	clk	P	
		fbs_wready	O	clk	P	
		fbs_bid[P_FBS_AXID_WIDTH-1:0]	O	clk	P	The port width is specified by a parameter.
		fbs_bresp[1:0]	O	clk	P	
		fbs_bvalid	O	clk	P	
		fbs_bready	I	clk	P	
		fbs_arid[P_FBS_AXID_WIDTH-1:0]	I	clk	P	The port width is specified by a parameter.
		fbs_araddr[31:0]	I	clk	P	
		fbs_arlen[7:0]	I	clk	P	
		fbs_arsize[2:0]	I	clk	P	
		fbs_arburst[1:0]	I	clk	P	
		fbs_arlock	I	clk	P	
		fbs_arcache[3:0]	I	clk	P	
		fbs_arprot[2:0]	I	clk	P	
		fbs_arvalid	I	clk	P	
		fbs_arready	O	clk	P	
		fbs_aruser	I	clk	P	1: Debug, 0: Non-Debug
		fbs_rid[P_FBS_AXID_WIDTH-1:0]	O	clk	P	The port width is specified by a parameter.
		fbs_rdata[31:0]	O	clk	P	
		fbs_rresp[1:0]	O	clk	P	
		fbs_rlast	O	clk	P	
		fbs_rvalid	O	clk	P	
		fbs_rready	I	clk	P	

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
4	Front Port Bus Slave (Cont.)	fbs_awcpayload[20:0]	I	clk	P	EDC protection code for write address channel payload [20:14] Code for fbs_awid[P_FBS_AXID_WIDTH-1:0] [13:7] Code for fbs_awlen[7:0], fbs_awsize[2:0], fbs_awburst[1:0], fbs_awlock, fbs_awcache[3:0], fbs_awprot[2:0], fbs_awuser [6:0] Code for fbs_awaddr[31:0]
		fbs_awcvalid	I	clk	P	EDC protection code for fbs_awvalid
		fbs_awcready	O	clk	P	EDC protection code for fbs_awready
		fbs_wcpayload[13:0]	I	clk	P	EDC protection code for write data channel payload [13:7] Code for fbs_wlast,fbs_wstrb[3:0] [6:0] Code for fbs_wdata[31:0]
		fbs_wcvalid	I	clk	P	EDC protection code for fbs_wvalid
		fbs_wcready	O	clk	P	EDC protection code for fbs_wready
		fbs_bcpayload[6:0]	O	clk	P	EDC protection code for write response channel payload [6:0] Code for fbs_bid[P_FBS_AXID_WIDTH-1:0], fbs_bresp[1:0]
		fbs_bcvalid	O	clk	P	EDC protection code for fbs_bvalid
		fbs_bcready	I	clk	P	EDC protection code for fbs_bready
		fbs_arcpayload[20:0]	I	clk	P	EDC protection code for read address channel payload [20:14] Code for fbs_arid[P_FBS_AXID_WIDTH-1:0] [13:7] Code for fbs_arlen[7:0], fbs_arsize[2:0], fbs_arburst[1:0], fbs_arlock, fbs_arcache[3:0], fbs_arprot[2:0], fbs_aruser [6:0] Code for fbs_araddr[31:0]
		fbs_arcvalid	I	clk	P	EDC protection code for fbs_arvalid
		fbs_arcready	O	clk	P	EDC protection code for fbs_arready
		fbs_rcpayload[13:0]	O	clk	P	EDC protection code for read data channel payload [13:7] Code for fbs_rid[P_FBS_AXID_WIDTH-1:0], fbs_rresp[1:0], fbs_rlast [6:0] Code for fbs_rdata[31:0]
		fbs_rcvalid	O	clk	P	EDC protection code for fbs_rvalid
		fbs_rcready	I	clk	P	EDC protection code for fbs_rready

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
5	Peripheral Bus Master	pbm_paddr[31:0]	O	clk	P	
		pbm_pprot[2:0]	O	clk	P	
		pbm_psel	O	clk	P	
		pbm_penable	O	clk	P	
		pbm_pwrite	O	clk	P	
		pbm_pwdata[31:0]	O	clk	P	
		pbm_pstrb[3:0]	O	clk	P	
		pbm_pready	I	clk	P	
		pbm_prdata[31:0]	I	clk	P	
		pbm_pslverr	I	clk	P	
		pbm_copayload[20:0]	O	clk	P	EDC protection code for output payload [20:14] Code for pbm_pprot[2 : 0], pbm_psel, pbm_penable, pbm_pwrite, pbm_pstrb[3 : 0] [13:7] Code for pbm_paddr[31:0] [6:0] Code for pbm_pwdata[31:0]
		pbm_cipayload[6:0]	I	clk	P	EDC protection code for input payload [6:0] Code for pbm_prdata[31:0]
		pbm_cpready	I	clk	P	EDC protection code for pbm_pready
		pbm_cpslverr	I	clk	P	EDC protection code for pbm_pslverr
6	Interrupt	int_extint[P_EXTINT_WIDTH:1]	I	clk	P	The External interrupt and port width are specified by parameters.
		int_extnmi	I	clk	P	External NMI
		int_errint	I	clk	P	Error Interrupt (maskable)
		int_errnmi	I	clk	P	Error NMI
7	JTAG	jtag_tck	I	-	-	TCK
		jtag_trst_n	I	jtag_tck	N	TRST (negative)
		jtag_tms	I	jtag_tck	P	TMS
		jtag_tdi	I	jtag_tck	P	TDI
		jtag_tdo	O	jtag_tck	P	TDO
		jtag_tdo_en	O	jtag_tck	P	TDO enable
8	Debug	dbg_auth	I	clk	P	Debug Enable
		dbg_halt_outreq	O	clk	P	Halt req to the out of NS31AUIL
		dbg_halt_outack	I	clk	P	Halt ack from the out of NS31AUIL
		dbg_resume_outreq	O	clk	P	Resume req to the out of NS31AUIL
		dbg_resume_outack	I	clk	P	Resume ack from the out of NS31AUIL
		dbg_halt_inreq	I	clk	P	Halt req from the out of NS31AUIL
		dbg_halt_inack	O	clk	P	Halt ack to the out of NS31AUIL
		dbg_resume_inreq	I	clk	P	Resume req from the out of NS31AUIL
		dbg_resume_inack	O	clk	P	Resume ack to the out of NS31AUIL

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
9	NS31A System Control	ncc_stat[3:0]	O	clk	P	System Status Output
		ncc_mststop_req	I	clk	P	State change request to "Master Stop"
		ncc_mststop_ack	O	clk	P	State change acknowledge to "Master Stop"
		ncc_sysstop_req	I	clk	P	State change request to "System Stop"
		ncc_sysstop_ack	O	clk	P	State change acknowledge to "System Stop"
		ncc_active_req	I	clk	P	State change request to "Active"
		ncc_active_ack	O	clk	P	State change acknowledge to "Active"
		ncc_sysrst_req	O	clk	P	System reset request
		ncc_sysrst_ack	I	clk	P	System reset acknowledge
		ncc_dmactive_req	O	clk	P	Power control suppression request
		ncc_dmactive_ack	I	clk	P	Power control suppression acknowledge
		ncc_cache_clr_req	I	clk	P	Cache clear request
		ncc_cache_clr_ack	O	clk	P	Cache clear acknowledge
		ncc_runhart	I	clk	P	Run Hart
		ncc_pmpcfgul	I	clk	P	PMPCFG.L unlockable
		ncc_mtime_cnten	I	clk	P	ACLINT.mtime count enable
10	Configuration	cfg_resetvec[31:0]	I	clk	P	Reset vector address
		cfg_nmivec[31:0]	I	clk	P	NMI vector address
		cfg_mtvec[31:0]	I	clk	P	Initial value of mtvec
		cfg_boothart	I	clk	P	Initial value of boothart
11	DFT	dft_disable_gck	I	clk	P	Clock gater disable
		dft_bypass_clk_inv	I	DC	P	Clock inverter bypass
		dft_bypass_int_rst	I	DC	P	Internal reset generator bypass
		dft_mask_reset	I	DC	P	Synchronous reset mask
		rst_dft	I	async	P	DFT reset

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
12	Error	err_pmp	O	clk	P	PMP Error
		err_bus_decerr	O	clk	P	Bus Error (Decode Error)
		err_bus_prterr	O	clk	P	Bus Error (Protocol Error)
		err_bus_slverr	O	clk	P	Bus Error (Slave Error)
		err_bus_sederr[2:0]	O	clk	P	Bus ECC 1-bit error [0] Front bus slave [1] System bus master [2] Ext. Peripheral APB
		err_bus_dederr[2:0]	O	clk	P	Bus ECC 2-bit error [0] Front bus slave [1] System bus master [2] Ext. Peripheral APB
		err_bus_ptyerr[2:0]	O	clk	P	Bus Parity error [0] Front bus slave [1] System bus master [2] Ext. Peripheral APB
		inj_bus_sederr[5:0]	I	clk	P	Bus 1-bit error injection [0] Front bus slave output data [1] System bus master output data [2] Ext. Peripheral APB output data [3] Front bus slave input data [4] System bus master input data [5] Ext. Peripheral APB input data
		inj_bus_dederr[5:0]	I	clk	P	Bus 2-bit error injection [0] Front bus slave output data [1] System bus master output data [2] Ext. Peripheral APB output data [3] Front bus slave input data [4] System bus master input data [5] Ext. Peripheral APB input data
		inj_bus_ptyerr[5:0]	I	clk	P	Bus Parity error injection [0] Front bus slave output data [1] System bus master output data [2] Reserved(Unused) [3] Front bus slave input data [4] System bus master input data [5] Ext. Peripheral APB input data

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
12	Error (Cont.)	err_dcls	O	clk	P	DCLS error
		inj_dcls_err	I	clk	P	DCLS error injection
		err_ram_sederr[3:0]	O	clk	P	RAM ECC 1-bit error [0] Inst. cache tag [1] Inst. cache data [2] DLM [3] ILM
		err_ram_dederr[3:0]	O	clk	P	RAM ECC 2-bit error [0] Inst. cache tag [1] Inst. cache data [2] DLM [3] ILM
		inj_ram_sederr[3:0]	I	clk	P	RAM 1-bit error injection [0] Inst. cache tag [1] Inst. cache data [2] DLM [3] ILM
		inj_ram_dederr[3:0]	I	clk	P	RAM 2-bit error injection [0] Inst. cache tag [1] Inst. cache data [2] DLM [3] ILM

12.2.2. NS31A CX Port List (AXI Configuration)

The following table lists the pins in the CX layer in the NS31A AXI configuration.

Table 197. NS31A CX Port List (AXI Configuration)

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
1	Clock	clk	I	-	-	System Clock
2	Reset	rst_pon	I	clk	P	Power-on Reset
		rst_sys	I	clk	P	System Reset

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
3	System Bus Master	sbm_awid[4:0]	O	clk	P	
		sbm_awaddr[31:0]	O	clk	P	
		sbm_awlen[7:0]	O	clk	P	
		sbm_awsize[2:0]	O	clk	P	
		sbm_awburst[1:0]	O	clk	P	
		sbm_awlock	O	clk	P	
		sbm_awcache[3:0]	O	clk	P	
		sbm_awprot[2:0]	O	clk	P	
		sbm_awvalid	O	clk	P	
		sbm_awready	I	clk	P	
		sbm_awuser	O	clk	P	1: Debug, 0: Non-Debug
		sbm_wdata[31:0]	O	clk	P	
		sbm_wstrb[3:0]	O	clk	P	
		sbm_wlast	O	clk	P	
		sbm_wvalid	O	clk	P	
		sbm_wready	I	clk	P	
		sbm_bid[4:0]	I	clk	P	
		sbm_bresp[1:0]	I	clk	P	
		sbm_bvalid	I	clk	P	
		sbm_bready	O	clk	P	
		sbm_arid[4:0]	O	clk	P	
		sbm_araddr[31:0]	O	clk	P	
		sbm_arlen[7:0]	O	clk	P	
		sbm_arsize[2:0]	O	clk	P	
		sbm_arburst[1:0]	O	clk	P	
		sbm_arlock	O	clk	P	
		sbm_arcache[3:0]	O	clk	P	
		sbm_arprot[2:0]	O	clk	P	
		sbm_arvalid	O	clk	P	
		sbm_arready	I	clk	P	
		sbm_aruser	O	clk	P	1: Debug, 0: Non-Debug
		sbm_rid[4:0]	I	clk	P	
		sbm_rdata[31:0]	I	clk	P	
		sbm_rresp[1:0]	I	clk	P	
		sbm_rlast	I	clk	P	
		sbm_rvalid	I	clk	P	
		sbm_rready	O	clk	P	

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
4	Front Port Bus Slave	fbs_awid[P_FBS_AXID_WIDTH-1:0]	I	clk	P	The port width is specified by a parameter.
		fbs_awaddr[31:0]	I	clk	P	
		fbs_awlen[7:0]	I	clk	P	
		fbs_awsize[2:0]	I	clk	P	
		fbs_awburst[1:0]	I	clk	P	
		fbs_awlock	I	clk	P	
		fbs_awcache[3:0]	I	clk	P	
		fbs_awprot[2:0]	I	clk	P	
		fbs_awvalid	I	clk	P	
		fbs_awready	O	clk	P	
		fbs_awuser	I	clk	P	1: Debug, 0: Non-Debug
		fbs_wdata[31:0]	I	clk	P	
		fbs_wstrb[3:0]	I	clk	P	
		fbs_wlast	I	clk	P	
		fbs_wvalid	I	clk	P	
		fbs_wready	O	clk	P	
		fbs_bid[P_FBS_AXID_WIDTH-1:0]	O	clk	P	The port width is specified by a parameter.
		fbs_bresp[1:0]	O	clk	P	
		fbs_bvalid	O	clk	P	
		fbs_bready	I	clk	P	
		fbs_arid[P_FBS_AXID_WIDTH-1:0]	I	clk	P	The port width is specified by a parameter.
		fbs_araddr[31:0]	I	clk	P	
		fbs_arlen[7:0]	I	clk	P	
		fbs_arsize[2:0]	I	clk	P	
		fbs_arburst[1:0]	I	clk	P	
		fbs_arlock	I	clk	P	
		fbs_arcache[3:0]	I	clk	P	
		fbs_arprot[2:0]	I	clk	P	
		fbs_arvalid	I	clk	P	
		fbs_arready	O	clk	P	
		fbs_aruser	I	clk	P	1: Debug, 0: Non-Debug
		fbs_rid[P_FBS_AXID_WIDTH-1:0]	O	clk	P	The port width is specified by a parameter.
		fbs_rdata[31:0]	O	clk	P	
		fbs_rresp[1:0]	O	clk	P	
		fbs_rlast	O	clk	P	
		fbs_rvalid	O	clk	P	
		fbs_rready	I	clk	P	

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
5	Peripheral Bus Master	pbm_paddr[31:0]	O	clk	P	
		pbm_pprot[2:0]	O	clk	P	
		pbm_psel	O	clk	P	
		pbm_penable	O	clk	P	
		pbm_pwrite	O	clk	P	
		pbm_pwdata[31:0]	O	clk	P	
		pbm_pstrb[3:0]	O	clk	P	
		pbm_pready	I	clk	P	
		pbm_prdata[31:0]	I	clk	P	
		pbm_pslverr	I	clk	P	
6	Interrupt	int_extint[P_EXTINT_WIDTH:1]	I	clk	P	The External interrupt and port width are specified by parameters.
		int_extnmi	I	clk	P	External NMI
		int_errint	I	clk	P	Error Interrupt (maskable)
		int_errnmi	I	clk	P	Error NMI
7	Debug	dbg_auth	I	clk	P	Debug Enable
8	NS31A System Control	ncc_stat[3:0]	O	clk	P	System Status Output
		ncc_mststop_req	I	clk	P	State change request to "Master Stop"
		ncc_mststop_ack	O	clk	P	State change acknowledge to "Master Stop"
		ncc_sysstop_req	I	clk	P	State change request to "System Stop"
		ncc_sysstop_ack	O	clk	P	State change acknowledge to "System Stop"
		ncc_active_req	I	clk	P	State change request to "Active"
		ncc_active_ack	O	clk	P	State change acknowledge to "Active"
		ncc_cache_clr_req	I	clk	P	Cache clear request
		ncc_cache_clr_ack	O	clk	P	Cache clear acknowledge
		ncc_runhart	I	clk	P	Run Hart
		ncc_pmpcfgul	I	clk	P	PMPCFG.L unlockable
		ncc_mtime_cnten	I	clk	P	ACLINT.mtime count enable
9	Error	err_pmp	O	clk	P	PMP Error
		err_bus_decerr	O	clk	P	Bus Decode Error
		err_bus_slverr	O	clk	P	Bus Slave Error
		err_bus_prterr	O	clk	P	Bus Protocol Error
10	Configuration	cfg_resetvec[31:0]	I	clk	P	Reset vector address
		cfg_nmivec[31:0]	I	clk	P	NMI vector address
		cfg_mtvec[31:0]	I	clk	P	Initial value of mtvec
		cfg_boothart	I	clk	P	Initial value of boothart
11	DFT	dft_disable_gck	I	clk	P	Clock gater disable
		dft_mask_reset	I	DC	P	Synchronous reset mask
12	DBG - Clock&Reset	db31a_clk	O	-	-	DBG Clock
		db31a_rst	O	clk	P	DBG Power On Reset
		db31a_rst_sys	O	clk	P	DBG System Reset

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
13	DBG - APB Master	dbs_paddr[31:0]	O	clk	P	
		dbs_pprot[2:0]	O	clk	P	
		dbs_psel	O	clk	P	
		dbs_penable	O	clk	P	
		dbs_pwrite	O	clk	P	
		dbs_pwdata[31:0]	O	clk	P	
		dbs_pstrb[3:0]	O	clk	P	
		dbs_pready	I	clk	P	
		dbs_prdata[31:0]	I	clk	P	
		dbs_pslverr	I	clk	P	
14	DBG - APB Slave	dbg_paddr[31:0]	I	clk	P	
		dbg_pprot[2:0]	I	clk	P	
		dbg_psel	I	clk	P	
		dbg_penable	I	clk	P	
		dbg_pwrite	I	clk	P	
		dbg_pwdata[31:0]	I	clk	P	
		dbg_pstrb[3:0]	I	clk	P	
		dbg_pready	O	clk	P	
		dbg_prdata[31:0]	O	clk	P	
		dbg_pslverr	O	clk	P	
15	DBG - Core Control	db31a_haltreq_hart	I	clk	P	
		db31a_halted_hart	O	clk	P	
		db31a_resumereq_hart	I	clk	P	
		db31a_resumed_hart	O	clk	P	
		dbg_ac_postexec_req_hart	I	clk	P	
		dbg_ac_postexec_ack_hart	O	clk	P	
		dbg_ac_cmderr_excp_hart	O	clk	P	
16	DBG - Core Status	db31a_status_hart_dbg	O	clk	P	
		halted_anyhart	I	clk	P	

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
17	MEM - IC	ic_tag_cen[1:0]	O	clk	P	For 2 ways
		ic_tag_wen[1:0]	O	clk	P	For 2 ways
		ic_tag_a[13:5]	O	clk	P	Maximum IC size
		ic_tag_d[43:0]	O	clk	P	For 2 ways (22bit x 2)
		ic_tag_q[43:0]	I	clk	P	For 2 ways (22bit x 2)
		ic_tag_q_corr[43:0]	I	clk	P	ECC-corrected data
		ic_tag_q_err_any	I	clk	P	Any error notification (regardless of SEC, SED, or DED)
		ic_tag_q_err_fatal	I	clk	P	Fatal error notification (SED/DED notification; excluding SEC)
		ic_tag_validate_ecc	O	clk	P	ECC error validation judgment for IC Tag "1" ECC error is valid. "0" ECC error is not valid (pseudo error).
		ic_data_cen[1:0]	O	clk	P	For 2 ways
		ic_data_wen[1:0]	O	clk	P	For 2 ways
		ic_data_a[13:3]	O	clk	P	Maximum IC size (2 ways)
		ic_data_d[127:0]	O	clk	P	For 2 ways (64bit x 2)
		ic_data_q[127:0]	I	clk	P	For 2 ways (64bit x 2)
		ic_data_q_corr[127:0]	I	clk	P	ECC-corrected data
		ic_data_q_err_any[1:0]	I	clk	P	Any error notification (regardless of SEC, SED, or DED), For 2 ways
		ic_data_q_err_fatal[1:0]	I	clk	P	Fatal error notification (SED/DED notification; excluding SEC), For 2 ways
		ic_data_validate_ecc[1:0]	O	clk	P	[0]ECC error validation judgment for IC Data Bank0 [1] ECC error validation judgment for IC Data Bank1 “1”: ECC error is valid. “0”: ECC error is not valid (pseudo error).
18	MEM - ILM	ilm_cen	O	clk	P	
		ilm_wen[1:0]	O	clk	P	1: higher bank, 0: lower bank
		ilm_a[22:3]	O	clk	P	Maximum ILM size (8 MB)
		ilm_d[63:0]	O	clk	P	[63:32]: higher bank, [31:0]: lower bank
		ilm_q[63:0]	I	clk	P	[63:32]: higher bank, [31:0]: lower bank
		ilm_q_corr[63:0]	I	clk	P	ECC-corrected data, [63:32]: higher bank, [31:0]: lower bank
		ilm_q_err_any	I	clk	P	Any error notification (regardless of SEC, SED, or DED)
		ilm_q_err_fatal	I	clk	P	Fatal error notification (SED/DED notification; excluding SEC)

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
19	MEM - DLM	dlm_cen	O	clk	P	
		dlm_wen	O	clk	P	
		dlm_a[22:2]	O	clk	P	Maximum DLM size (8 MB)
		dlm_d[31:0]	O	clk	P	
		dlm_q[31:0]	I	clk	P	
		dlm_q_corr[31:0]	I	clk	P	ECC-corrected data
		dlm_q_err_any	I	clk	P	Any error notification (regardless of SEC, SED, or DED)
		dlm_q_err_fatal	I	clk	P	Fatal error notification (SED/DED notification; excluding SEC)

12.2.3. NS31A DBG Port List (AXI Configuration)

The following table lists the pins in the DBG layer in the NS31A AXI configuration.

Table 198. DBG Port List (AXI Configuration)

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
1	System	clk	I	-	-	Clock
		rst_dbg	I	clk	P	Debug reset (sync)
		rst_sys	I	clk	P	System reset
		dbg_auth	I	clk	P	Debug Authentication
2	DFT	rst_dft	I	clk	P	DFT reset (async)
		dft_bypass_clk_inv	I	-	P	Clock inverter bypass (DC)
		dft_bypass_int_rst	I	-	P	Internal reset generator bypass (DC)
3	JTAG Interface	jtag_tck	I	-	-	TCK
		jtag_trst_n	I	-	N	TRST (negative)
		jtag_tms	I	jtag_tck	P	TMS
		jtag_tdi	I	jtag_tck	P	TDI
		jtag_tdo	O	jtag_tck(neg)	P	TDO
		jtag_tdo_en	O	jtag_tck(neg)	P	TDO enable (1 : JTAG_TDO OUTPUT)

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
4	DM Register Interface (APB4 slave)	dbs_paddr[PaddrWidth-1:0]	I	clk	P	dm IF(APB) Paddr
		dbs_pprot[2:0]	I	clk	P	dm IF(APB) Pprot
		dbs_psel	I	clk	P	dm IF(APB) Psel
		dbs_penable	I	clk	P	dm IF(APB) Penable
		dbs_pwrite	I	clk	P	dm IF(APB) Pwrite
		dbs_pwdata[31:0]	I	clk	P	dm IF(APB) Pwdata
		dbs_pstrb[3:0]	I	clk	P	dm IF(APB) Pstrb
		dbs_pauser	I	clk	P	Fix to 0
		dbs_pready	O	clk	P	dm IF(APB) Pready
		dbs_prdata[31:0]	O	clk	P	dm IF(APB) Prdata
		dbs_pslverr	O	clk	P	dm IF(APB) Pslverr
5	External Trigger	dbg_halt_inreq	I	clk	P	Halt Request
		dbg_halt_inack	O	clk	P	Halt Acknoledge
		dbg_resume_inreq	I	clk	P	Resume Request
		dbg_resume_inack	O	clk	P	Resume Acknoledge
		dbg_halt_outreq	O	clk	P	Halt Request
		dbg_halt_outack	I	clk	P	Halt Acknoledge
		dbg_resume_outreq	O	clk	P	Resume Request
		dbg_resume_outack	I	clk	P	Resume Acknoledge
6	Halt Control	haltreq_hart[HtNum-1:0]	O	clk	P	Halt request
		halted_hart[HtNum-1:0]	I	clk	P	Halt complete
		resumereq_hart[HtNum-1:0]	O	clk	P	Resume request
		resumed_hart[HtNum-1:0]	I	clk	P	Resume complete
		status_hart_debug[HtNum-1:0]	I	clk	P	debug mode status
		halted_any_hart	O	clk	P	Hart halt status
7	Abstract Control	dbg_ac_req	O	clk	P	Abstract Command request
		dbg_ac_transfer	O	clk	P	command.transfer
		dbg_ac_postexec	O	clk	P	commnad.postexec
		dbg_ac_write	O	clk	P	commnad.write
		dbg_ac_aarsize[2:0]	O	clk	P	commnad.aarsize
		dbg_ac_regno[15:0]	O	clk	P	commnad.regno
		dbg_ac_wdata[63:0]	O	clk	P	data1,data0
		dbg_ac_hartsel[19:0]	O	clk	P	dmcontrol.harselhi hartsello
		dbg_ac_progbuf[63:0]	O	clk	P	progbuf1,progbug0
		dbg_ac_ack	I	clk	P	Abstract Command acknowledge
		dbg_ac_rdata[63:0]	I	clk	P	Read data
		dbg_ac_cmderr[2:0]	I	clk	P	Error Abstractcs.cmderr

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
8	Reset Control	ndmreset_req	O	clk	P	system reset request
		ndmreset_ack	I	clk	P	system reset acknowledge
		dmactive_req	O	clk	P	DM activation request
		dmactive_ack	I	clk	P	DM activation acknowledge

12.2.4. NS31A MEM Port List (AXI Configuration)

The following table lists the ports in the MEM layer in the NS31A AXI configuration.

Table 199. MEM Port List (AXI Configuration)

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
1	Clock&Reset	clk	I	-	-	
2	IC	ic_tag_cen[1:0]	I	clk	P	For 2 ways
		ic_tag_wen[1:0]	I	clk	P	For 2 ways
		ic_tag_a[13:5]	I	clk	P	Maximum IC size
		ic_tag_d[43:0]	I	clk	P	For 2 ways (22 bits x 2)
		ic_tag_q[43:0]	O	clk	P	For 2 ways (22 bits x 2)
		ic_tag_ram_rd[1:0]	O	clk	P	IC (tag) read data valid (2 ways)
		ic_data_cen[1:0]	I	clk	P	For 2 ways
		ic_data_wen[1:0]	I	clk	P	For 2 ways
		ic_data_a[13:3]	I	clk	P	Maximum IC size (2 ways)
		ic_data_d[127:0]	I	clk	P	For 2 ways (64 bits x 2)
		ic_data_q[127:0]	O	clk	P	For 2 ways (64 bits x 2)
		ic_data_ram_rd[1:0]	O	clk	P	IC (data) read data valid (2 ways)
3	ILM	ilm_cen	I	clk	P	
		ilm_wen_h	I	clk	P	For upper 32 bits
		ilm_wen_l	I	clk	P	For lower 32 bits
		ilm_a[22:3]	I	clk	P	Maximum ILM size (8 MB)
		ilm_d[63:0]	I	clk	P	
		ilm_q[63:0]	O	clk	P	
		ilm_ram_rd	O	clk	P	ILM read data valid
4	DLM	dlm_cen	I	clk	P	
		dlm_wen	I	clk	P	
		dlm_a[22:2]	I	clk	P	Maximum DLM size (8 MB)
		dlm_d[31:0]	I	clk	P	
		dlm_q[31:0]	O	clk	P	
		dlm_ram_rd	O	clk	P	DLM read data valid

No	Type	Port Name	Direction	Synch.	Polarity	Function (Comment)
5	ECC	ic_tag_q_cpayload[13:0]	O	clk	P	ECC for IC (tag) read data
		ic_data_q_cpayload[15:0]	O	clk	P	ECC for IC (data) read data
		ilm_q_cpayload[13:0]	O	clk	P	ECC for ILM read data
		dlm_q_cpayload[6:0]	O	clk	P	ECC for DLM read data
		ic_tag_d_cpayload[13:0]	I	clk	P	ECC for IC (tag) write data
		ic_data_d_cpayload[15:0]	I	clk	P	ECC for IC (data) write data
		ilm_d_cpayload[13:0]	I	clk	P	ECC for ILM write data
		dlm_d_cpayload[6:0]	I	clk	P	ECC for DLM write data

12.2.5. NS31A UIL Port List (AHB Configuration)

The following table lists the pins in the UIL layer in the NS31A AHB configuration.

Table 200. NS31A UIL Port List (AHB Configuration)

No	Type	Port Name	Direction	Synchronization	Polarity	Function (Comment)
1	Clock	-	-	-	-	Same as the AXI configuration [*1]
2	Reset	-	-	-	-	Same as the AXI configuration [*1]
3	Instruction Bus Master	ibm_haddr[31:0]	O	clk	P	
		ibm_hburst[2:0]	O	clk	P	3’b000: SINGLE 3’b101: INCR8
		ibm_hprot[3:0]	O	clk	P	Instruction Bus Master protocol [0] 0 fixed [1] 0:User, 1:Privileged [3:2] P_IBM_HPROT3_2_FVAL
		ibm_hsize[2:0]	O	clk	P	
		ibm_htrans[1:0]	O	clk	P	
		ibm_hauser	O	clk	P	1:Debug, 0:Non-debug
		ibm_hrdata[31:0]	I	clk	P	
		ibm_hready	I	clk	P	
		ibm_hresp	I	clk	P	

No	Type	Port Name	Direction	Synchronization	Polarity	Function (Comment)
4	Data Bus Master	dbm_haddr[31:0]	O	clk	P	
		dbm_hburst[2:0]	O	clk	P	3'b000: Fixed to SINGLE
		dbm_hmastlock	O	clk	P	
		dbm_hprot[3:0]	O	clk	P	Data Bus Master protocol [0] 1 fixed [1] 0:User, 1:Privileged [3:2] P_DBM_HPROT3_2_FVAL
		dbm_hsize[2:0]	O	clk	P	
		dbm_hexcl	O	clk	P	
		dbm_htrans[1:0]	O	clk	P	
		dbm_hwdata[31:0]	O	clk	P	
		dbm_hwrite	O	clk	P	
		dbm_hauser	O	clk	P	1:Debug, 0:Non-debug
		dbm_hrdata[31:0]	I	clk	P	
		dbm_hready	I	clk	P	
		dbm_hresp	I	clk	P	
		dbm_hexokay	I	clk	P	
5	Peripheral Bus Slave	pbs_paddr[31:0]	I	clk	P	
		pbs_pprot[2:0]	I	clk	P	
		pbs_psel	I	clk	P	
		pbs_penable	I	clk	P	
		pbs_pwrite	I	clk	P	
		pbs_pwdata[31:0]	I	clk	P	
		pbs_pstrb[3:0]	I	clk	P	
		pbs_pready	O	clk	P	
		pbs_prdata[31:0]	O	clk	P	
		pbs_pslverr	O	clk	P	
6	Interrupt	-	-	-	-	Same as the AXI configuration [*1]
7	JTAG	-	-	-	-	Same as the AXI configuration [*1]
8	Debug	-	-	-	-	Same as the AXI configuration [*1]
9	NS31A System Control	-	-	-	-	Same as the AXI configuration [*1]
10	Configuration	-	-	-	-	Same as the AXI configuration [*1]
11	DFT	-	-	-	-	Same as the AXI configuration [*1]

No	Type	Port Name	Direction	Synchronization	Polarity	Function (Comment)
12	Error	err_pmp	O	clk	P	
		err_dcls	O	clk	P	
		inj_dcls_err	I	clk	P	
		err_ram_sederr[1:0]	O	clk	P	RAM ECC 1-bit error [0] Inst. Cache tag [1] Inst. Cache data
		err_ram_dederr[1:0]	O	clk	P	RAM ECC 2-bit error [0] Inst. Cache tag [1] Inst. Cache data
		inj_ram_sederr[1:0]	I	clk	P	RAM 1-bit error injection [0] Inst. Cache tag [1] Inst. Cache data
		inj_ram_dederr[1:0]	I	clk	P	RAM 2-bit error injection [0] Inst. Cache tag [1] Inst. Cache data

1. See Table 196.

12.2.6. NS31A CX Port List (AHB Configuration)

The following table lists the pins in the CX layer in the NS31A AHB configuration.

Table 201. NS31A CX Port List (AHB Configuration)

No	Type	Port Name	Direction	Synchronization	Polarity	Function (Comment)
1	Clock	-	-	-	-	Same as the AXI configuration [*1]
2	Reset	-	-	-	-	Same as the AXI configuration [*1]
3	Instruction Bus Master	ibm_haddr[31:0]	O	clk	P	
		ibm_hburst[2:0]	O	clk	P	3’b000: Fixed to SINGLE 3’b101: INCR8
		ibm_hprot[3:0]	O	clk	P	Instruction Bus Master protocol [0] 0 fixed [1] 0:User, 1:Privileged [3:2] P_IBM_HPROT3_2_FVAL
		ibm_hsize[2:0]	O	clk	P	
		ibm_htrans[1:0]	O	clk	P	
		ibm_hauser	O	clk	P	1:Debug, 0:Non-debug
		ibm_hrdata[31:0]	I	clk	P	
		ibm_hready	I	clk	P	
		ibm_hresp	I	clk	P	

No	Type	Port Name	Direction	Synchronization	Polarity	Function (Comment)
4	Data Bus Master	dbm_haddr[31:0]	O	clk	P	
		dbm_hburst[2:0]	O	clk	P	3’b000: Fixed to SINGLE
		dbm_hmastlock	O	clk	P	
		dbm_hprot[3:0]	O	clk	P	Data Bus Master protocol [0] 1 fixed [1] 0:User, 1:Privileged [3:2] P_DBM_HPROT3_2_FVAL
		dbm_hsize[2:0]	O	clk	P	
		dbm_hexcl	O	clk	P	
		dbm_htrans[1:0]	O	clk	P	
		dbm_hwdata[31:0]	O	clk	P	
		dbm_hwrite	O	clk	P	
		dbm_hauser	O	clk	P	1:Debug, 0:Non-debug
		dbm_hrdata[31:0]	I	clk	P	
		dbm_hready	I	clk	P	
		dbm_hresp	I	clk	P	
		dbm_hexokay	I	clk	P	
5	Peripheral Bus Slave	pbs_paddr[31:0]	I	clk	P	
		pbs_pprot[2:0]	I	clk	P	
		pbs_psel	I	clk	P	
		pbs_penable	I	clk	P	
		pbs_pwrite	I	clk	P	
		pbs_pwdata[31:0]	I	clk	P	
		pbs_pstrb[3:0]	I	clk	P	
		pbs_pready	O	clk	P	
		pbs_prdata[31:0]	O	clk	P	
		pbs_pslverr	O	clk	P	
6	Interrupt	-	-	-	-	Same as the AXI configuration *1
7	Debug	-	-	-	-	Same as the AXI configuration *1
8	NS31ASystemControl	-	-	-	-	Same as the AXI configuration *1
9	Error	err_pmp	O	clk	P	PMPError
10	Configuration	-	-	-	-	Same as the AXI configuration *1
11	DFT	-	-	-	-	Same as the AXI configuration *1
12	DBG-Clock&Reset	-	-	-	-	Same as the AXI configuration *1

No	Type	Port Name	Direction	Synchronization	Polarity	Function (Comment)
13	DBG-APB Master	db31a_m_paddr[31:0]	O	clk	P	Change the pin name.
		db31a_m_pprot[2:0]	O	clk	P	Change the pin name.
		db31a_m_psel	O	clk	P	Change the pin name.
		db31a_m_penable	O	clk	P	Change the pin name.
		db31a_m_pwrite	O	clk	P	Change the pin name.
		db31a_m_pwdata[31:0]	O	clk	P	Change the pin name.
		db31a_m_pstrb[3:0]	O	clk	P	Change the pin name.
		db31a_m_pready	I	clk	P	Change the pin name.
		db31a_m_prdata[31:0]	I	clk	P	Change the pin name.
		db31a_m_pslverr	I	clk	P	Change the pin name.
14	DBG-APB Slave	db31a_s_paddr[31:0]	I	clk	P	TL-→APB
		db31a_s_pprot[2:0]	I	clk	P	TL-→APB
		db31a_s_psel	I	clk	P	TL-→APB
		db31a_s_penable	I	clk	P	TL-→APB
		db31a_s_pwrite	I	clk	P	TL-→APB
		db31a_s_pwdata[31:0]	I	clk	P	TL-→APB
		db31a_s_pstrb[3:0]	I	clk	P	TL-→APB
		db31a_s_pauser	I	clk	P	TL-→APB
		db31a_s_pready	O	clk	P	TL-→APB
		db31a_s_prdata[31:0]	O	clk	P	TL-→APB
		db31a_s_pslverr	O	clk	P	TL-→APB
15	DBG-StatusControl	-	-	-	-	Same as the AXI configuration *1
16	DBG-CoreControl	-	-	-	-	Same as the AXI configuration *1
17	DBG-CoreStatus	-	-	-	-	Same as the AXI configuration *1
18	MEM-IC	-	-	-	-	Same as the AXI configuration *1
19	MEM-ILM	All deleted*2	-	-	-	
20	MEM-DLM	All deleted*2	-	-	-	
21	FuSa	ibm_read_data_phase_valid	O	clk	P	Control Signal for DCLS
		dbm_data_phase_valid	O	clk	P	Control Signal for DCLS
		dbm_read_data_phase_valid	O	clk	P	Control Signal for DCLS
		dbm_write_data_phase_valid	O	clk	P	Control Signal for DCLS

1. See Table 197.
2. The ports were deleted because ILM and DLM were removed from UIL.

12.2.7. NS31A DBG Port List (AHB Configuration)

The pins in the DBG layer in the NS31A AHB configuration are the same as those in the AXI configuration. See Table 198.

12.2.8. NS31A MEM Port List (AHB Configuration)

The following table lists the pins in the MEM layer in the NS31A AHB configuration.

Table 202. MEM Port List (AHB Configuration)

No	Type	Port Name	Direction	Synchronization	Polarity	Function (Comment)
1	Clock&Reset	-	-	-	-	Same as the AXI configuration [*1]
2	IC	-	-	-	-	Same as the AXI configuration [*1]
3	ILM	All deleted[*2]	-	-	-	
4	DLM	All deleted[*2]	-	-	-	
5	ECC	ic_tag_q_cpayload[13:0]	O	clk	P	ECC for IC (tag) read data
		ic_data_q_cpayload[15:0]	O	clk	P	ECC for IC (data) read data
		ic_tag_d_cpayload[13:0]	I	clk	P	ECC for IC (tag) write data
		ic_data_d_cpayload[15:0]	I	clk	P	ECC for IC (data) write data

1. See Table 199.
2. The ports were deleted because ILM and DLM were removed from UIL.

Appendix A: NS31A CSR Register

This appendix lists the CSRs integrated in NS31A.

Table 203. NS31A CSR Register List

Register No.	Register Name	Register Descriptions
User Floating-Point CSRs		
001_H	fflags	Section 3.7.2.1.1
002_H	frm	Section 3.7.2.1.2
003_H	fcsr	Section 3.7.2.1.3
Machine Trap Setup		
300_H	mstatus	Section 3.7.2.2.1
301_H	misa	Section 3.7.2.2.2
304_H	mie	Section 3.7.2.2.3
305_H	mtvec	Section 3.7.2.2.4
306_H	mcounteren	Section 3.7.2.2.5
310_H	mstatush	Section 3.7.2.2.6
Machine Configuration		
$30A_H$	menvcfg	Section 3.7.2.3.1
$31A_H$	menvcfgh	Section 3.7.2.3.2
Machine Counter Setup		
320_H	mcountinhibit	Section 3.7.2.4.1
Machine Trap Handling		
340_H	mscratch	Section 3.7.2.5.1
341_H	mepc	Section 3.7.2.5.2
342_H	mcause	Section 3.7.2.5.3
343_H	mtval	Section 3.7.2.5.4
344_H	mip	Section 3.7.2.5.5
Machine Non-Maskable Interrupt Handling		
740_H	mnscratch	Section 3.7.2.6.1
741_H	mnepc	Section 3.7.2.6.2
742_H	mncause	Section 3.7.2.6.3
744_H	mnstatus	Section 3.7.2.6.4
Machine Memory Protection		
$3A0_H$	pmpcfg0	Section 3.7.2.7.1
$3A1_H$	pmpcfg1	Section 3.7.2.7.1
$3A2_H$	pmpcfg2	Section 3.7.2.7.1
$3A3_H$	pmpcfg3	Section 3.7.2.7.1
$3B0_H$	pmpaddr0	Section 3.7.2.7.2
$3B1_H$	pmpaddr1	Section 3.7.2.7.2
$3B2_H$	pmpaddr2	Section 3.7.2.7.2
$3B3_H$	pmpaddr3	Section 3.7.2.7.2

Register No.	Register Name	Register Descriptions
$3B4_H$	pmpaddr4	Section 3.7.2.7.2
$3B5_H$	pmpaddr5	Section 3.7.2.7.2
$3B6_H$	pmpaddr6	Section 3.7.2.7.2
$3B7_H$	pmpaddr7	Section 3.7.2.7.2
$3B8_H$	pmpaddr8	Section 3.7.2.7.2
$3B9_H$	pmpaddr9	Section 3.7.2.7.2
$3BA_H$	pmpaddr10	Section 3.7.2.7.2
$3BB_H$	pmpaddr11	Section 3.7.2.7.2
$3BC_H$	pmpaddr12	Section 3.7.2.7.2
$3BD_H$	pmpaddr13	Section 3.7.2.7.2
$3BE_H$	pmpaddr14	Section 3.7.2.7.2
$3BF_H$	pmpaddr15	Section 3.7.2.7.2
Debug/Trace Registers (shared with Debug Mode)		
$7A0_H$	tselect	Section 3.7.2.8.1
$7A1_H$	tdata1	Section 3.7.2.8.2
$7A2_H$	tdata2	Section 3.7.2.8.3
$7A4_H$	tinfo	Section 3.7.2.8.4
Debug Mode Registers		
$7B0_H$	dcsr	Section 3.7.2.9.1
$7B1_H$	dpc	Section 3.7.2.9.2
$7B2_H$	dscratch0	Section 3.7.2.9.3
Machine Counter/Timers		
$B00_H$	mcycle	Section 3.7.2.10.1
$B02_H$	minstret	Section 3.7.2.10.2
$B80_H$	mcycleh	Section 3.7.2.10.4
$B82_H$	minstreth	Section 3.7.2.10.5
User Counter/Timers		
$C00_H$	cycle	Section 3.7.2.11.1
$C01_H$	time	Section 3.7.2.11.2
$C02_H$	instret	Section 3.7.2.11.3
$C80_H$	cycleh	Section 3.7.2.11.5
$C81_H$	timeh	Section 3.7.2.11.6
$C82_H$	instreth	Section 3.7.2.11.7
Machine Information Registers		
$F11_H$	mvendorid	Section 3.7.2.12.1
$F12_H$	marchid	Section 3.7.2.12.2
$F13_H$	mimpid	Section 3.7.2.12.3
$F14_H$	mhartid	Section 3.7.2.12.4
$F15_H$	mconfigptr	Section 3.7.2.12.5

Appendix B: NS31A Internal Register

This appendix lists the registers integrated in the Internal Register Region except for the Debug Subsystem Region of NS31A.

Table 204. NS31A Internal Register List

Region	Base Address	Offset Address	Register Symbol	Descriptions
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+0000$_H$	msip	Section 7.2.2.1
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+4000$_H$	mtimecmp	Section 7.2.2.2
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+4004$_H$	mtimecmph	Section 7.2.2.3
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+D000$_H$	mtimectrl	Section 7.2.2.4
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+BFF8$_H$	mtime	Section 7.2.2.5
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+BFFC$_H$	mtimeh	Section 7.2.2.6
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+FF00$_H$	mswicfg	Section 7.2.2.7
ACLINT Control Register Region	P_ADDR_BASE_ACLINT	+FF40$_H$	mtimercfg	Section 7.2.2.8
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000000$_H$	eispri1	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000004$_H$	eispri2	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000008$_H$	eispri3	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00000C$_H$	eispri4	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000010$_H$	eispri5	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000014$_H$	eispri6	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000018$_H$	eispri7	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00001C$_H$	eispri8	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000020$_H$	eispri9	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000024$_H$	eispri10	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000028$_H$	eispri11	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00002C$_H$	eispri12	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000030$_H$	eispri13	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000034$_H$	eispri14	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000038$_H$	eispri15	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00003C$_H$	eispri16	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000040$_H$	eispri17	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000044$_H$	eispri18	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000048$_H$	eispri19	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00004C$_H$	eispri20	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000050$_H$	eispri21	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000054$_H$	eispri22	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000058$_H$	eispri23	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00005C$_H$	eispri24	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000060$_H$	eispri25	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000064$_H$	eispri26	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000068$_H$	eispri27	Section 8.2.2.1

Region	Base Address	Offset Address	Register Symbol	Descriptions
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00006C$_H$	eispri28	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000070$_H$	eispri29	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000074$_H$	eispri30	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000078$_H$	eispri31	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00007C$_H$	eispri32	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000080$_H$	eispri33	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000084$_H$	eispri34	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000088$_H$	eispri35	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00008C$_H$	eispri36	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000090$_H$	eispri37	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000094$_H$	eispri38	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000098$_H$	eispri39	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00009C$_H$	eispri40	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000A0$_H$	eispri41	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000A4$_H$	eispri42	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000A8$_H$	eispri43	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000AC$_H$	eispri44	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000B0$_H$	eispri45	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000B4$_H$	eispri46	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000B8$_H$	eispri47	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000BC$_H$	eispri48	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000C0$_H$	eispri49	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000C4$_H$	eispri50	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000C8$_H$	eispri51	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000CC$_H$	eispri52	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000D0$_H$	eispri53	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000D4$_H$	eispri54	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000D8$_H$	eispri55	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000DC$_H$	eispri56	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000E0$_H$	eispri57	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000E4$_H$	eispri58	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000E8$_H$	eispri59	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000EC$_H$	eispri60	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000F0$_H$	eispri61	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000F4$_H$	eispri62	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000F8$_H$	eispri63	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0000FC$_H$	eispri64	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000100$_H$	eispri65	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000104$_H$	eispri66	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000108$_H$	eispri67	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00010C$_H$	eispri68	Section 8.2.2.1

Region	Base Address	Offset Address	Register Symbol	Descriptions
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000110$_H$	eispri69	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000114$_H$	eispri70	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000118$_H$	eispri71	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00011C$_H$	eispri72	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000120$_H$	eispri73	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000124$_H$	eispri74	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000128$_H$	eispri75	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00012C$_H$	eispri76	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000130$_H$	eispri77	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000134$_H$	eispri78	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000138$_H$	eispri79	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00013C$_H$	eispri80	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000140$_H$	eispri81	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000144$_H$	eispri82	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000148$_H$	eispri83	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00014C$_H$	eispri84	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000150$_H$	eispri85	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000154$_H$	eispri86	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000158$_H$	eispri87	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00015C$_H$	eispri88	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000160$_H$	eispri89	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000164$_H$	eispri90	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000168$_H$	eispri91	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00016C$_H$	eispri92	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000170$_H$	eispri93	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000174$_H$	eispri94	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000178$_H$	eispri95	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00017C$_H$	eispri96	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000180$_H$	eispri97	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000184$_H$	eispri98	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000188$_H$	eispri99	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00018C$_H$	eispri100	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000190$_H$	eispri101	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000194$_H$	eispri102	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000198$_H$	eispri103	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00019C$_H$	eispri104	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001A0$_H$	eispri105	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001A4$_H$	eispri106	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001A8$_H$	eispri107	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001AC$_H$	eispri108	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001B0$_H$	eispri109	Section 8.2.2.1

Region	Base Address	Offset Address	Register Symbol	Descriptions
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001B4$_H$	eispri110	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001B8$_H$	eispri111	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001BC$_H$	eispri112	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001C0$_H$	eispri113	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001C4$_H$	eispri114	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001C8$_H$	eispri115	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001CC$_H$	eispri116	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001D0$_H$	eispri117	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001D4$_H$	eispri118	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001D8$_H$	eispri119	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001DC$_H$	eispri120	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001E0$_H$	eispri121	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001E4$_H$	eispri122	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001E8$_H$	eispri123	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001EC$_H$	eispri124	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001F0$_H$	eispri125	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001F4$_H$	eispri126	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001F8$_H$	eispri127	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0001FC$_H$	eispri128	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000200$_H$	eispri129	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000204$_H$	eispri130	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000208$_H$	eispri131	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00020C$_H$	eispri132	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000210$_H$	eispri133	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000214$_H$	eispri134	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000218$_H$	eispri135	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00021C$_H$	eispri136	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000220$_H$	eispri137	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000224$_H$	eispri138	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000228$_H$	eispri139	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00022C$_H$	eispri140	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000230$_H$	eispri141	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000234$_H$	eispri142	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000238$_H$	eispri143	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00023C$_H$	eispri144	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000240$_H$	eispri145	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000244$_H$	eispri146	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000248$_H$	eispri147	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00024C$_H$	eispri148	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000250$_H$	eispri149	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000254$_H$	eispri150	Section 8.2.2.1

Region	Base Address	Offset Address	Register Symbol	Descriptions
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000258$_H$	eispri151	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00025C$_H$	eispri152	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000260$_H$	eispri153	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000264$_H$	eispri154	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000268$_H$	eispri155	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00026C$_H$	eispri156	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000270$_H$	eispri157	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000274$_H$	eispri158	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000278$_H$	eispri159	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00027C$_H$	eispri160	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000280$_H$	eispri161	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000284$_H$	eispri162	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000288$_H$	eispri163	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00028C$_H$	eispri164	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000290$_H$	eispri165	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000294$_H$	eispri166	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000298$_H$	eispri167	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00029C$_H$	eispri168	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002A0$_H$	eispri169	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002A4$_H$	eispri170	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002A8$_H$	eispri171	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002AC$_H$	eispri172	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002B0$_H$	eispri173	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002B4$_H$	eispri174	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002B8$_H$	eispri175	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002BC$_H$	eispri176	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002C0$_H$	eispri177	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002C4$_H$	eispri178	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002C8$_H$	eispri179	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002CC$_H$	eispri180	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002D0$_H$	eispri181	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002D4$_H$	eispri182	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002D8$_H$	eispri183	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002DC$_H$	eispri184	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002E0$_H$	eispri185	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002E4$_H$	eispri186	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002E8$_H$	eispri187	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002EC$_H$	eispri188	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002F0$_H$	eispri189	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002F4$_H$	eispri190	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002F8$_H$	eispri191	Section 8.2.2.1

Region	Base Address	Offset Address	Register Symbol	Descriptions
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0002FC$_H$	eispri192	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000300$_H$	eispri193	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000304$_H$	eispri194	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000308$_H$	eispri195	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00030C$_H$	eispri196	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000310$_H$	eispri197	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000314$_H$	eispri198	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000318$_H$	eispri199	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00031C$_H$	eispri200	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000320$_H$	eispri201	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000324$_H$	eispri202	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000328$_H$	eispri203	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00032C$_H$	eispri204	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000330$_H$	eispri205	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000334$_H$	eispri206	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000338$_H$	eispri207	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00033C$_H$	eispri208	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000340$_H$	eispri209	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000344$_H$	eispri210	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000348$_H$	eispri211	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00034C$_H$	eispri212	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000350$_H$	eispri213	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000354$_H$	eispri214	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000358$_H$	eispri215	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00035C$_H$	eispri216	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000360$_H$	eispri217	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000364$_H$	eispri218	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000368$_H$	eispri219	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00036C$_H$	eispri220	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000370$_H$	eispri221	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000374$_H$	eispri222	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000378$_H$	eispri223	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00037C$_H$	eispri224	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000380$_H$	eispri225	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000384$_H$	eispri226	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000388$_H$	eispri227	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00038C$_H$	eispri228	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000390$_H$	eispri229	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000394$_H$	eispri230	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+000398$_H$	eispri231	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00039C$_H$	eispri232	Section 8.2.2.1

Region	Base Address	Offset Address	Register Symbol	Descriptions
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003A0$_H$	eispri233	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003A4$_H$	eispri234	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003A8$_H$	eispri235	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003AC$_H$	eispri236	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003B0$_H$	eispri237	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003B4$_H$	eispri238	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003B8$_H$	eispri239	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003BC$_H$	eispri240	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003C0$_H$	eispri241	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003C4$_H$	eispri242	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003C8$_H$	eispri243	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003CC$_H$	eispri244	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003D0$_H$	eispri245	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003D4$_H$	eispri246	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003D8$_H$	eispri247	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003DC$_H$	eispri248	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003E0$_H$	eispri249	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003E4$_H$	eispri250	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003E8$_H$	eispri251	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003EC$_H$	eispri252	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003F0$_H$	eispri253	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003F4$_H$	eispri254	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+0003F8$_H$	eispri255	Section 8.2.2.1
PLIC Control Register Region	P_ADDR_BASE_PLIC	+001000$_H$	eispen0	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+001004$_H$	eispen1	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+001008$_H$	eispen2	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00100C$_H$	eispen3	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+001010$_H$	eispen4	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+001014$_H$	eispen5	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+001018$_H$	eispen6	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00101C$_H$	eispen7	Section 8.2.2.2
PLIC Control Register Region	P_ADDR_BASE_PLIC	+002000$_H$	eihen0	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+002004$_H$	eihen1	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+002008$_H$	eihen2	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00200C$_H$	eihen3	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+002010$_H$	eihen4	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+002014$_H$	eihen5	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+002018$_H$	eihen6	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+00201C$_H$	eihen7	Section 8.2.2.3
PLIC Control Register Region	P_ADDR_BASE_PLIC	+1F8000$_H$	eisfstat0	Section 8.2.2.4
PLIC Control Register Region	P_ADDR_BASE_PLIC	+1F8004$_H$	eisfstat1	Section 8.2.2.4

Region	Base Address	Offset Address	Register Symbol	Descriptions
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F8008_H$	eisfstat2	Section 8.2.2.4
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F800C_H$	eisfstat3	Section 8.2.2.4
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F8010_H$	eisfstat4	Section 8.2.2.4
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F8014_H$	eisfstat5	Section 8.2.2.4
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F8018_H$	eisfstat6	Section 8.2.2.4
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F801C_H$	eisfstat7	Section 8.2.2.4
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F9000_H$	eisfclr0	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F9004_H$	eisfclr1	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F9008_H$	eisfclr2	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F900C_H$	eisfclr3	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F9010_H$	eisfclr4	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F9014_H$	eisfclr5	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F9018_H$	eisfclr6	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1F901C_H$	eisfclr7	Section 8.2.2.5
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC000_H$	eicfgst0	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC004_H$	eicfgst1	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC008_H$	eicfgst2	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC00C_H$	eicfgst3	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC010_H$	eicfgst4	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC014_H$	eicfgst5	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC018_H$	eicfgst6	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FC01C_H$	eicfgst7	Section 8.2.2.6
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+1FD000_H$	eicfghi	Section 8.2.2.7
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+200000_H$	eihth	Section 8.2.2.8
PLIC Control Register Region	P_ADDR_BASE_PLIC	$+200004_H$	eihcid	Section 8.2.2.9
CMU Control Register Region	P_ADDR_BASE_CMU	$+0200_H$	STATSYS	Section 6.2.2.1
CMU Control Register Region	P_ADDR_BASE_CMU	$+0210_H$	STATCORE	Section 6.2.2.2
CMU Control Register Region	P_ADDR_BASE_CMU	$+0300_H$	STATHTRUN	Section 6.2.2.3
CMU Control Register Region	P_ADDR_BASE_CMU	$+0304_H$	STATHTDBG	Section 6.2.2.4
CMU Control Register Region	P_ADDR_BASE_CMU	$+0308_H$	STATHTDIS	Section 6.2.2.5
CMU Control Register Region	P_ADDR_BASE_CMU	$+0310_H$	HARTRUN	Section 6.2.2.6
CMU Control Register Region	P_ADDR_BASE_CMU	$+0320_H$	HARTDIS	Section 6.2.2.7
CMU Control Register Region	P_ADDR_BASE_CMU	$+0400_H$	STATCCLR	Section 6.2.2.8
CMU Control Register Region	P_ADDR_BASE_CMU	$+0410_H$	CTRLCCLR	Section 6.2.2.9
CMU Control Register Region	P_ADDR_BASE_CMU	$+0600_H$	FUNCCTRL	Section 6.2.2.10
CMU Control Register Region	P_ADDR_BASE_CMU	$+0700_H$	SSCRATCH0	Section 6.2.2.11
CMU Control Register Region	P_ADDR_BASE_CMU	$+0704_H$	SSCRATCH1	Section 6.2.2.11
CMU Control Register Region	P_ADDR_BASE_CMU	$+0800_H$	MONDBG	Section 6.2.2.12
CMU Control Register Region	P_ADDR_BASE_CMU	$+0804_H$	MONRST	Section 6.2.2.13
CMU Control Register Region	P_ADDR_BASE_CMU	$+0808_H$	MONSYS	Section 6.2.2.14
CMU Control Register Region	P_ADDR_BASE_CMU	$+080C_H$	MONINT	Section 6.2.2.15

Region	Base Address	Offset Address	Register Symbol	Descriptions
CMU Control Register Region	P_ADDR_BASE_CMU	+0900$_H$	BOOTHART	Section 6.2.2.16
CMU Control Register Region	P_ADDR_BASE_CMU	+0904$_H$	INITMTVEC	Section 6.2.2.17
CMU Control Register Region	P_ADDR_BASE_CMU	+0908$_H$	INITRSTVEC	Section 6.2.2.18
CMU Control Register Region	P_ADDR_BASE_CMU	+0910$_H$	INITNMIVEC	Section 6.2.2.19
CMU Control Register Region	P_ADDR_BASE_CMU	+0E00$_H$	CFGPMPNUM	Section 6.2.2.20
CMU Control Register Region	P_ADDR_BASE_CMU	+0E04$_H$	CFGCSIZEIC	Section 6.2.2.21
CMU Control Register Region	P_ADDR_BASE_CMU	+0FF0$_H$	MARCHID	Section 6.2.2.22
CMU Control Register Region	P_ADDR_BASE_CMU	+0FF4$_H$	MIMPID	Section 6.2.2.23
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0000$_H$	EXCLMONVALIDBITENTRY0	Section 3.8.3.1.1
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0004$_H$	EXCLMONADDRESSENTRY0	Section 3.8.3.1.2
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0008$_H$	EXCLMONIDENTRY0	Section 3.8.3.1.3
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+000C$_H$	EXCLMONSIDEBANDENTRY0	Section 3.8.3.1.4
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0010$_H$	EXCLMONVALIDBITENTRY1	Section 3.8.3.1.1
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0014$_H$	EXCLMONADDRESSENTRY1	Section 3.8.3.1.2
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0018$_H$	EXCLMONIDENTRY1	Section 3.8.3.1.3
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+001C$_H$	EXCLMONSIDEBANDENTRY1	Section 3.8.3.1.4
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0100$_H$	LINKBIT0	Section 3.8.3.1.5
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0104$_H$	LINKADDR0	Section 3.8.3.1.6
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0600$_H$	L1CRACTRL	Section 3.8.3.1.7
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0604$_H$	L1CRATGT	Section 3.8.3.1.8
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0608$_H$	L1CRADAT0	Section 3.8.3.1.9
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+060C$_H$	L1CRADAT1	Section 3.8.3.1.10
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0800$_H$	BTBACTRL	Section 3.8.3.1.11
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0804$_H$	BTBATGT	Section 3.8.3.1.12
Memory Mapped Control Register Region	P_ADDR_BASE_REG	+0808$_H$	BTBADAT	Section 3.8.3.1.13
Hart Context Register Region	P_ADDR_BASE_HCR	+1F0C$_H$	meecaddr	Section 3.8.3.2.1
Hart Context Register Region	P_ADDR_BASE_HCR	+1F10$_H$	meecunlock	Section 3.8.3.2.2
Hart Context Register Region	P_ADDR_BASE_HCR	+2F00$_H$	mhtpc	Section 3.8.3.2.3
Hart Context Register Region	P_ADDR_BASE_HCR	+2F40$_H$	mhtenable	Section 3.8.3.2.4

Region	Base Address	Offset Address	Register Symbol	Descriptions
Hart Context Register Region	P_ADDR_BASE_HCR	+2F80$_H$	mhtstatus	Section 3.8.3.2.5
Hart Context Register Region	P_ADDR_BASE_HCR	+2FA0$_H$	ml1cachesysctrl	Section 3.8.3.2.6
Hart Context Register Region	P_ADDR_BASE_HCR	+2FA4$_H$	ml1cacheoperation	Section 3.8.3.2.7
Hart Context Register Region	P_ADDR_BASE_HCR	+2FA8$_H$	ml1cacheprefcfg	Section 3.8.3.2.8
Hart Context Register Region	P_ADDR_BASE_HCR	+2FB8$_H$	mbpsysctrl	Section 3.8.3.2.9
Hart Context Register Region	P_ADDR_BASE_HCR	+2FBC$_H$	mbpoperation	Section 3.8.3.2.10
Hart Context Register Region	P_ADDR_BASE_HCR	+3F00$_H$	mhtfetchaddr	Section 3.8.3.2.11

Appendix C: Instruction Table

This appendix shows the list of instructions supported by NS31A.

C.1. Legend

[Mnemonic]

The mnemonic of the instruction call.

[Description]

A brief description of the instruction behavior.

[Format]

The instruction behavior is described using pseudo code.
The meaning of each operand in the pseudo code is described below.

[Comment]

The supplementary explanation of the instruction operands is provided below.

Table 205. Operand List

Operand	Description
rd	Represents a destination register. In the case of a floating-point instruction, this represents a floating-point register.
rs1, rs2	Represents a source register. In the case of a floating-point instruction, this represents a floating-point register. When written within parentheses, as in (rs1), this represents a memory address. If "'" is suffixed in an RV32C instruction such as "rs2'", only x8 to x15 can be specified.
rd/rs1	Represents a source register and a destination register alternately in an RV32C instruction. If "'" is suffixed in an RV32C instruction such as "rd'/rs1'", only x8 to x15 can be specified.
sp	Represents only x2 (sp) can be specified as operand in an RV32C instruction. When written within parentheses, as in (sp), this represents a memory address.
imm	Signed immediate value 12 bits
imm6	Signed immediate value 6 bits
imm20	Signed immediate value 20 bits
uimm	Unsigned immediate value 12 bits
uimm5	Unsigned immediate value 5 bits
nzimm6	Signed immediate value 6 bits other than 0
nzuimm8	Unsigned immediate value 8 bits other than 0
shamt	Shift amount (5 bits for RV32)
offset	Represents an offset value (signed 12 bits)
offset5	Represents an offset value (unsigned 5 bits)
offset6	Represents an offset value (unsigned 6 bits)
offset8	Represents an offset value (signed 8 bits)
offset11	Represents an offset value (signed 11 bits)
offset20	Represents an offset value (signed 20 bits)
csr	Represents the Control and Status Register (CSR).

C.2. RV32I, Zicsr, Zifencei, SYSTEM

The RV32I, Zicsr, Zifencei, and SYSTEM instructions supported by NS31A are shown.

- RV32I instructions: Table 206
- Zicsr instructions: Table 207
- Zifencei instructions: Table 208
- SYSTEM instructions: Table 209

Table 206. RV32I Instruction List

Mnemonic	Description	Format	Comment
Integer Register-Register Operations			
add	Add	add rd, rs1, rs2	
sub	Subtract	sub rd, rs1, rs2	
sll	Shift Left Logical	sll rd, rs1, rs2	Only lower 5 bits in rs2 are used for the shift amount.
slt	Set if Less Than	slt rd, rs1, rs2	
sltu	Set if Less Than, Unsigned	sltu rd, rs1, rs2	
xor	Exclusive-OR	xor rd, rs1, rs2	
srl	Shift Right Logical	srl rd, rs1, rs2	Only lower 5 bits in rs2 are used for the shift amount.
sra	Shift Right Arithmetic	sra rd, rs1, rs2	Only lower 5 bits in rs2 are used for the shift amount.
or	OR	or rd, rs1, rs2	
and	AND	and rd, rs1, rs2	
Integer Register-Immediate Operations			
slli	Shift Left Logical Immediate	slli rd, rs1, shamt	
srli	Shift Right Logical Immediate	srli rd, rs1, shamt	
srai	Shift Right Arithmetic Immediate	srai rd, rs1, shamt	
addi	Add Immediate	addi rd, rs1, imm	
slti	Set if Less Than Immediate	slti rd, rs1, imm	
sltiu	Set if Less Than Immediate, Unsigned	sltiu rd, rs1, imm	imm is treated as unsigned after sign extension.
xori	Exclusive-OR Immediate	xori rd, rs1, imm	
ori	OR Immediate	ori rd, rs1, imm	
andi	AND Immediate	andi rd, rs1, imm	
lui	Load Upper Immediate	lui rd, imm20	imm20 is treated after being left-shifted by 12 bits.
auipc	Add Upper Immediate to PC	auipc rd, offset20	offset20 is treated after being left-shifted by 12 bits.
Control Transfer			
jal	Jump and Link	jal rd, offset20	offset20 is treated as 21-bit data after being added with 0 to the lowest bit.
jalr	Jump and Link Register	jalr rd, rs1, offset20	offset20 is treated as 21-bit data after being added with 0 to the lowest bit.
Conditional Branches			
beq	Branch if Equal	beq rs1, rs2, offset	offset is treated as 13-bit data after being added with 0 to the lowest bit.

Mnemonic	Description	Format	Comment
bne	Branch if Not Equal	bne rs1, rs2, offset	offset is treated as 13-bit data after being added with 0 to the lowest bit.
blt	Branch if Less Than	blt rs1, rs2, offset	offset is treated as 13-bit data after being added with 0 to the lowest bit.
bge	Branch if Greater Than or Equal	bge rs1, rs2, offset	offset is treated as 13-bit data after being added with 0 to the lowest bit.
bltu	Branch if Less Than, Unsigned	bltu rs1, rs2, offset	offset is treated as 13-bit data after being added with 0 to the lowest bit.
bgeu	Branch if Greater Than or Equal, Unsigned	bgeu rs1, rs2, offset	offset is treated as 13-bit data after being added with 0 to the lowest bit.
Load and Store			
lb	Load Byte	lb rd, offset(rs1)	
lh	Load Halfword	lh rd, offset(rs1)	
lw	Load Word	lw rd, offset(rs1)	
lbu	Load Byte, Unsigned	lbu rd, offset(rs1)	
lhu	Load Halfword, Unsigned	lhu rd, offset(rs1)	
sb	Store Byte	sb rs2, offset(rs1)	
sh	Store Halfword	sh rs2, offset(rs1)	
sw	Store Word	sw rs2, offset(rs1)	
Memory Ordering			
fence	Fence Memory and I/O	fence	NS31A does not treat predecessor and successor.

Table 207. Zicsr Instruction List

Mnemonic	Description	Format	Comment
Control and Status Register			
csrrw	Control and Status Register Read and Write	csrrw rd, csr, rs1	
csrrs	Control and Status Register Read and Set	csrrs rd, csr, rs1	
csrrc	Control and Status Register Read and Clear	csrrc rd, csr, rs1	
csrrwi	Control and Status Register Read and Write Immediate	csrrwi rd, csr, uimm5	
csrrsi	Control and Status Register Read and Set Immediate	csrrsi rd, csr, uimm5	
csrrci	Control and Status Register Read and Clear Immediate	csrrci rd, csr, uimm5	

Table 208. Zifencei Instruction List

Mnemonic	Description	Format	Comment
Instruction-Fetch Fence			
fence.i	Fence Instruction Stream	fence.i	

Table 209. SYSTEM Instruction List

Mnemonic	Description	Format	Comment
Machine-mode Privileged Instruction			
ecall	Environment Call	ecall	
ebreak	Environment Breakpoint	ebreak	
mret	Machine-mode Exception Return	mret	
mnret	Machine-mode NMI Return	mnret	
wfi	Wait for Interrupt	wfi	

C.3. RV32M

Table 210 shows the RV32M instructions supported by NS31A.

Table 210. RV32M Instruction List

Mnemonic	Description	Format	Comment
Multipication & Division			
mul	Multiply	mul rd, rs1, rs2	
mulh	Multiply High	mulh rd, rs1, rs2	
mulhsu	Multiply High Signed-Unsigned	mulhsu rd, rs1, rs2	
mulhu	Multiply High Unsigned	mulhu rd, rs1, rs2	
div	Divide	div rd, rs1, rs2	
divu	Divide, Unsigned	divu rd, rs1, rs2	
rem	Remainder	rem rd, rs1, rs2	
remu	Remainder, Unsigned	remu rd, rs1, rs2	

C.4. RV32A

Table 211 shows the RV32A instructions supported by NS31A.

Table 211. RV32A Instruction List

Mnemonic	Description	Format	Comment
Load-Reserved/Store-Conditional			
lr.w	Load-Reserved Word	lr.w rd, (rs1)	
sc.w	Store-Conditional Word	sc.w rd, rs2, (rs1)	
Atomic Memory Operations			
amoswap.w	Atomic Memory Operation: Swap Word	amoswap.w rd, rs2, (rs1)	
amoadd.w	Atomic Memory Operation: Add Word	amoadd.w rd, rs2, (rs1)	
amoxor.w	Atomic Memory Operation: XOR Word	amoxor.w rd, rs2, (rs1)	
amoand.w	Atomic Memory Operation: AND Word	amoand.w rd, rs2, (rs1)	
amoor.w	Atomic Memory Operation: OR Word	amoor.w rd, rs2, (rs1)	
amomin.w	Atomic Memory Operation: Minimum Word	amomin.w rd, rs2, (rs1)	
amomax.w	Atomic Memory Operation: Maximum Word	amomax.w rd, rs2, (rs1)	
amominu.w	Atomic Memory Operation: Minimum Word, Unsigned	amominu.w rd, rs2, (rs1)	
amomaxu.w	Atomic Memory Operation: Maximum Word, Unsigned	amomaxu.w rd, rs2, (rs1)	

C.5. RV32C

Table 212 shows the RV32C instructions supported by NS31A.

Table 212. RV32C instructions (compact) List

Mnemonic	Description	Format	Comment
Load and Store			
c.lwsp	Stack-Pointer-Based Load Word	c.lwsp rd, offset6(sp)	offset6 is treated as 8-bit data after being added with 00 to the lowest 2 bits. rd must not be x0.
c.flwsp	Stack-Pointer-Based Floating-point Load Word	c.flwsp rd, offset6(sp)	offset6 is treated as 8-bit data after being added with 00 to the lowest 2 bits.
c.swsp	Stack-Pointer-Based Store Word	c.swsp rs2, offset6(sp)	offset6 is treated as 8-bit data after being added with 00 to the lowest 2 bits.
c.fswsp	Stack-Pointer-Based Floating-point Store Word	c.fswsp rs2, offset6(sp)	offset6 is treated as 8-bit data after being added with 00 to the lowest 2 bits.
c.lw	Register-Based Load Word	c.lw rd', offset5(rs1')	offset5 is treated as 7-bit data after being added with 00 to the lowest 2 bits.
c.flw	Register-Based Floating-point Load Word	c.flw rd', offset5(rs1')	offset5 is treated as 7-bit data after being added with 00 to the lowest 2 bits.
c.sw	Register-Based Store Word	c.sw rs2', offset5(rs1')	offset5 is treated as 7-bit data after being added with 00 to the lowest 2 bits.
c.fsw	Register-Based Floating-point Store Word	c.fsw rs2', offset5(rs1')	offset5 is treated as 7-bit data after being added with 00 to the lowest 2 bits.
Control Transfer			
c.j	Jump	c.j offset11	offset11 is treated as 12-bit data after being added with 0 to the lowest bit.
c.jal	Jump and Link	c.jal offset11	offset11 is treated as 12-bit data after being added with 0 to the lowest bit.
c.jr	Jump Register	c.jr rs1	rs1 must not be x0.
c.jalr	Jump Register and Link	c.jalr rs1	rs1 must not be x0.
Conditional Branches			
c.beqz	Branch Equal Zero	c.beqz rs1', offset8	offset8 is treated as 9-bit data after being added with 0 to the lowest bit.
c.bnez	Branch not Equal Zero	c.bnez rs1', offset8	offset8 is treated as 9-bit data after being added with 0 to the lowest bit.
Integer Register-Immediate Operations			
c.li	Load Immediate	c.li rd, imm6	rd must not be x0.
c.lui	Load Upper Immediate	c.lui rd, nzimm6	rd must not be x0 and x2. nzimm6 is treated after being left-shifted by 12 bits.
c.addi	Add Immediate	c.addi rd/rs1, nzimm6	rd/rs1 must not be x0.
c.addi16sp	Add Immediate to Stack Pointer	c.addi16sp sp, nzimm6	nzimm6 is treated as 10-bit data after being added with 0000 to the lowest 4 bits.
c.addi4spn	Add Immediate to Stack Pointer	c.addi4spn rd', sp, nzuimm8	nzuimm8 is treated as 10-bit data after being added with 00 to the lowest 2 bits.
c.slli	Shift Left Logical Immediate	c.slli rd/rs1, shamt	rd/rs1 must not be x0.
c.srli	Shift Right Logical Immediate	c.srli rd'/rs1', shamt	
c.srai	Shift Right Arithmetical Immediate	c.srai rd'/rs1', shamt	

Mnemonic	Description	Format	Comment
c.andi	And Immediate	c.andi rd'/rs1', imm6	
Integer Register-Register Operations			
c.mv	Move	c.mv rd, rs2	rd and rs2 must not be x0.
c.add	Add	c.add rd/rs1, rs2	rd/rs1 and rs2 must not be x0.
c.and	AND	c.and rd'/rs1', rs2'	
c.or	OR	c.or rd'/rs1', rs2'	
c.xor	XOR	c.xor rd'/rs1', rs2'	
c.sub	Subtract	c.sub rd'/rs1', rs2'	
System			
c.unimp	Illegal Instruction	c.unimp	
c.nop	No Operation	c.nop	
c.ebreak	Environment Breakpoint	c.ebreak	

C.6. RV32F

This section shows the RV32F instructions supported by NS31A.

C.6.1. RV32F instructions (scalar floating-point instructions)

Table 213 shows the list of scalar floating-point instructions.

Table 213. RV32F Instruction (Scalar Floating-point Instruction) List

Mnemonic	Description	Format	Comment
Single-Precision Load and Store			
flw	Floating-point Load Word	flw rd, offset(rs1)	offset is treated as 13-bit data after being added with 0 to the lowest bit.
fsw	Floating-point Store Word	fsw rs2, offset(rs1)	offset is treated as 13-bit data after being added with 0 to the lowest bit.
Single-Precision Floating-Point Computational			
fmadd.s	Floating-point Fused Multiply-Add, Single-Precision	fmadd.s rd, rs1, rs2, rs3	
fmsub.s	Floating-point Fused Multiply-Subtract, Single-Precision	fmsub.s rd, rs1, rs2, rs3	
fnmsub.s	Floating-point Fused Negative Multiply-Subtract, Single-Precision	fnmsub.s rd, rs1, rs2, rs3	
fnmadd.s	Floating-point Fused Negative Multiply-Add, Single-Precision	fnmadd.s rd, rs1, rs2, rs3	
fadd.s	Floating-point Add, Single-Precision	fadd.s rd, rs1, rs2	
fsub.s	Floating-point Subtract, Single-Precision	fsub.s rd, rs1, rs2	
fmul.s	Floating-point Multiply, Single-Precision	fmul.s rd, rs1, rs2	
fdiv.s	Floating-point Divide, Single-Precision	fdiv.s rd, rs1, rs2	
fsgnj.s	Floating-point Sign Inject, Single-Precision	fsgnj.s rd, rs1, rs2	
fsgnjn.s	Floating-point Sign Inject-Negate, Single-Precision	fsgnjn.s rd, rs1, rs2	
fsgnjx.s	Floating-point Sign Inject-XOR, Single-Precision	fsgnjx.s rd, rs1, rs2	
fsqrt.s	Floating-point Square Root, Single-Precision	fsqrt.s rd, rs1	
fmin.s	Floating-point Minimum, Single-Precision	fmin.s rd, rs1, rs2	
fmax.s	Floating-point Maximum, Single-Precision	fmax.s rd, rs1, rs2	
Single-Precision Floating-Point Compare			
feq.s	Floating-point Equals, Single-Precision	feq.s rd, rs1, rs2	
fle.s	Floating-point Less Than or Equal, Single-Precision	fle.s rd, rs1, rs2	
flt.s	Floating-point Less Than, Single-Precision	flt.s rd, rs1, rs2	
Single-Precision Floating-Point Classify			
fclass.s	Floating-point Classify, Single-Precision	fclass.s rd, rs1	
Single-Precision Floating-Point Conversion and Move			

Mnemonic	Description	Format	Comment
fcvt.s.w	Floating-point Convert to Single from Word	fcvt.s.w rd, rs1	
fcvt.s.wu	Floating-point Convert to Single from Unsigned Word	fcvt.s.wu rd, rs1	
fcvt.w.s	Floating-point Convert to Word from Single	fcvt.w.s rd, rs1	
fcvt.wu.s	Floating-point Convert to Unsigned Word from Single	fcvt.wu.s rd, rs1	
fmv.x.w	Floating-point Move Word to Integer	fmv.x.w rd, rs1	
fmv.w.x	Floating-point Move Word from Integer	fmv.w.x rd, rs1	

C.6.2. RV32F instructions (NS31A unique additional instructions)

Table 214 shows the NS31A unique additional instructions.

Table 214. RV32F Instruction (NS31A Unique Additional Instruction) List

Mnemonic	Description	Format	Comment
NS31A custom instructions			
fexp2.s	Floating-Point Base-2 Exponential	fexp2.s rd, rs1	
flog2.s	Floating-Point Base-2 Logarithm	flog2.s rd, rs1	

C.7. RV32B

Table 215 shows the RV32B instructions supported by NS31A.

Table 215. RV32B Instruction List

Mnimonic	Description	Format	Comment
Zba extention			
sh1add	Shift left by 1 and add	sh1add rd, rs1, rs2	X(rd) = X(rs2) + (X(rs1) << 1);
sh2add	Shift left by 2 and add	sh2add rd, rs1, rs2	X(rd) = X(rs2) + (X(rs1) << 2);
sh3add	Shift left by 3 and add	sh3add rd, rs1, rs2	X(rd) = X(rs2) + (X(rs1) << 3);
Zbb: Basic bit-manipulation			
andn	AND with inverted operand	andn rd, rs1, rs2	X(rd) = X(rs1) & ~X(rs2);
orn	OR with inverted operand	orn rd, rs1, rs2	X(rd) = X(rs1) | ~X(rs2);
xnor	Exclusive NOR	xnor rd, rs1, rs2	X(rd) = ~(X(rs1) ^ X(rs2));
clz	Count leading zero bits	clz rd, rs	X(rd) = number of zeros of X(rs) from MSB
ctz	Count trailing zero bits	ctz rd, rs	X(rd) = number of zeros of X(rs) from LSB
cpop	Count set bits	cpop rd, rs	X(rd) = number of 1s of X(rs)
max	Maximum	max rd, rs1, rs2	X(rd) = max(X(rs1), X(rs2))
maxu	Unsigned maximum	maxu rd, rs1, rs2	X(rd) = maxu(X(rs1), X(rs2))
min	Minimum	min rd, rs1, rs2	X(rd) = min(X(rs1), X(rs2))
minu	Unsigned minimum	minu rd, rs1, rs2	X(rd) = minu(X(rs1), X(rs2))
sext.b	Sign-extend byte	sext.b rd, rs	X(rd) = 24{X(rs)[7]}.X(rs)[7:0]
sext.h	Sign-extend halfword	sext.h rd, rs	X(rd) = 16{X(rs)[15]}.X(rs)[15:0]
zext.h	Zero-extend halfword	zext.h rd, rs	X(rd) = 16{0}.X(rs)[15:0]
rol	Rotate left (Register)	rol rd, rs1, rs2	X(rd) = (X(rs1) << X(rs2)[4:0]) | (X(rs1) >> (XLEN - X(rs2)[4:0]))
ror	Rotate right (Register)	ror rd, rs1, rs2	X(rd) = (X(rs1) >> X(rs2)[4:0]) | (X(rs1) << (XLEN - X(rs2)[4:0]))
rori	Rotate right (Immediate)	rori rd, rs, shamt	X(rd) = (X(rs1) >> shamt) | (X(rs1) << (XLEN - shamt))
orc.b	Bitwise OR-Combine, byte granule	orc.b rd, rs	X(rd)[i*8+7:i*8] = 8{|X(rs)[7:0]} for i=0:3
rev8	Byte-reverse register	rev8 rd, rs	X(rd)[i*8+7:i*8] = X(rs)[(3-i)*8+7:(3-i)*8] for i=0:3
Zbs: Single-bit instruction			
bclr	Single-Bit Clear (Register)	bclr rd, rs1, rs2	X(rd) = X(rs1) & ~(1 << X(rs2)[4:0])
bclri	Single-Bit Clear (Immediater)	bclri rd, rs1, imm	X(rd) = X(rs1) & ~(1 << shamt)
bext	Single-Bit Extract (Register)	bext rd, rs1, rs2	X(rd) = (X(rs1) >> X(rs2)[4:0]) & 1
bexti	Single-Bit Extract (Immediater)	bexti rd, rs1, imm	X(rd) = (X(rs1) >> shamt) & 1
binv	Single-Bit Invert (Register)	binv rd, rs1, rs2	X(rd) = X(rs1) ^ (1 << X(rs2)[4:0])
binvi	Single-Bit Invert (Immediater)	binvi rd, rs1, imm	X(rd) = X(rs1) ^ (1 << shamt)
bset	Single-Bit Set (Register)	bset rd, rs1, rs2	X(rd) = X(rs1) | (1 << X(rs2)[4:0])
bseti	Single-Bit Set (Immediater)	bseti rd, rs1, imm	X(rd) = X(rs1) | (1 << shamt)

Appendix D: Instruction OpCode Table

See Reference Documents 1 and 2 for the opcode map of the NS31A instructions.
This appendix provides the opcode map of the unique instructions integrated in NS31A.
For the description of the legend, see Section C.1.

D.1. RV32F

D.1.1. RV32F (NS31A custom)

Table 216. RV32F OpCode Table (NS31A custom)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Floating-Point Base-2 Exponential																																
mnemonic	**funct5**					**fmt**		**rs2**					**rs1**					**rm**			**rd**					**opcode**						
fexp2.s	0	1	0	1	1	0	0	0	0	1	0	0	rs1					rm			rd					0	1	0	1	0	1	1
Floating-Point Base-2 Logarithm																																
mnemonic	**funct5**					**fmt**		**rs2**					**rs1**					**rm**			**rd**					**opcode**						
flog2.s	0	1	0	1	1	0	0	0	0	1	0	1	rs1					rm			rd					0	1	0	1	0	1	1

Revision History

Revision	Date	Description
1.0.0	September 22, 2021	Initial revision for public release
1.0.1	May 31, 2022	Release for description update
2.0.0	February 07, 2023	Release for description update
3.0.0	December 07, 2023	Release for description update

Change History

This section shows the history of technical changes. This history does not include the corrections of simple erroneous descriptions (e.g. typo). The following shows examples of titles.

[Additional specifications]

This is used when new specifications are added.

[Specification change]

This is used when there is a change to specifications described in this document. This is also used if the correction of an erroneous description may have an influence on the specifications.

[Supplementary explanation]

This is used when any supplementary description is added to help readers understand the specifications.

Differences between Revision 2.0.0 and Revision 3.0.0

Additional Specifications

- ACLINT and PLIC specifications were added to Reference Documents.
- In 1.2, ACLINT and PLIC were added to Functional Block Diagram.
- In 2.1.1.4, a description about P_BUS_PROT0_PRIV parameter was added.
- In 2.1.1.5, CMU Register Region Address Map was added.
- In 2.1.1.6, ACLINT Register Region Address Map was added.
- In 2.1.1.7, PLIC Register Region Address Map was added.
- In 2.1.1.8, a description about P_BUS_PROT0_PRIV parameter was added.
- In 2.1.1.9, a description about P_BUS_PROT0_PRIV parameter was added.
- In 2.1.2, additional space to each master’s accessible space was added.
- In 2.1.2.1, additional space was added to attributes per address area.
- In 3.7.2, mhpmevent3-10 were added to CSR Listing.
- In 3.7.2, mhpmcounter3-10 were added to CSR Listing.
- In 3.7.2, mhpmcounter3h-10h were added to CSR Listing.
- In 3.7.2, hpmcounter3-10 were added to CSR Listing.
- In 3.7.2, hpmcounter3h-10h were added to CSR Listing.
- In 3.7.2.2.5, a descrition about HPM3-10 was added to mcounteren.
- In 3.7.2.4.1, a description about HPM3-10 was added to mcountinhibit.
- In 3.7.2.4.2, a description about mhpmevent3-10 was added.

- In 3.7.2.10.3, a description about mhpmcounter3-10 was added.
- In 3.7.2.10.6, a description about mhpmcounter3h-10h was added.
- In 3.7.2.11.4, a description about hpmcounter3-10 was added.
- In 3.7.2.11.8, a description about hpmcounter3h-10h was added.
- In 3.7.3, a description about Hardware Performance Monitor (HPM) was added.
- In 3.10.1, ACLINT/PLIC/CMU Regions were added to AMO Region Table.
- In 3.10.2, ACLINT/PLIC/CMU Regions were added to LR/SC Region Table.
- In 4.1.1, a note on P_BUS_PROT0_PRIV was added to System Bus Master Spec Table.
- In 4.1.3, a note on P_BUS_PROT0_PRIV was added to Front Bus Slave Spec Table.
- Chapter 7 Advanced Core Local Interruptor (ACLINT) was added.
- Chapter 8 Platform-Level Interrupt Controller (PLIC) was added.
- 11.2.6 Control signals for DCLS were added in NS31ACX port list of AHB configuration.
- In 12.1, P_RV32B was added to Configuration Parameter List.
- In 12.1, P_HPMNUM was added to Configuration Parameter List.
- In 12.1, P_ADDR_BASE_ACLINT/PLIC/CMU were added to Configuration Parameter List.
- In 12.1, P_BUS_PROT0_PRIV was added to Configuration Parameter List.
- In 12.1, P_EXTINT_EDGE was added to Configuration Parameter List.
- In 12.2.1, ncc_mtime_cnten pin was added to NS31A UIL Port List.
- In 12.2.1, ncc_sysrst_ack pin was added to NS31A UIL Port List.
- In 12.2.1, ncc_dmactive_req/ack pin was added to NS31A UIL Port List.
- In 12.2.2, ncc_mtime_cnten pin was added to NS31A CX Port List.
- In 12.2.2, DBG-APB Slave pin was added to NS31A CX Port List.
- In 12.2.2, halted_anyhart was added to DBG-Core Status in NS31A CX Port List.

Specification Change

- Maximum capacity of ILM/DLM was changed to 8 MB.
    - 1.1 Features
    - 2.2. RAM
- 3.3.3 External interrupt acceptance condition was changed.
- 3.4 Target vector for Exceptions error interrupt in vector mode was changed, mtvec+40h → mtvec+4Ch.
- 3.7.2.11.2 Type of Flash Operation field was changed.
- 3.10.2 LR/SC operations were changed in AHB configuration

- 6.2.2.9 Initial value of CCLRREQ field of CTRLCCLR register was changed to 0.
    - 12.1. Configuration Parameter List
    - 12.2.2. NS31A CX_PortList (AXI Configuration)
    - 12.2.4. NS31A MEM Port List (AXI Configuration)
- ICU was deleted.
    - Reference Documents
    - 7
    - 7.1
    - 7.1.1
    - 7.1.2
    - 7.2
    - 7.2.1
    - 7.2.2
    - 7.2.2.1
    - 7.2.2.2
    - 7.2.2.3
    - 7.2.2.4
    - 7.2.2.5
    - 7.2.2.6
    - 7.2.2.7
    - 7.2.2.8
    - 7.2.2.9
    - 7.2.2.10
    - 7.2.2.11
    - 7.2.2.12
    - 7.2.2.13
    - 7.2.2.14
    - 7.2.2.15
    - 7.2.2.16
    - 7.2.2.17
    - 7.2.2.18
    - 7.2.2.19

- 7.2.2.20
- 7.2.2.21
- 7.2.2.22
- 7.2.2.23
- 7.2.2.24
- 7.3
- 7.3.1
- 7.3.2
- 7.3.3
- 7.3.3.1
- 7.3.3.1.1
- 7.3.3.1.2
- 7.3.3.2
- 7.3.3.2.1
- 7.3.3.2.2
- 7.3.3.3
- 7.3.3.3.1
- 7.3.3.3.2
- 7.3.3.4
- 7.3.3.4.1
- 7.3.3.4.2
- 7.3.4
- 7.3.5
- 7.3.6
- 7.3.7
- Appendix B

- Specifications and descriptions were changed according to the elimination of restrictions on BTB (BTB&RV32C).
  - In 3.8.3.1.13, the description about BTBADAT was changed.
  - In 12.1, the description about P_RV32C in Configuration Parameter List was changed.
  - In 12.1, the description about P_BTB in Configuration Parameter List was changed.
- Specifications and descriptions were changed for compliance with RISC-V External Debug Spec 1.0.0.
  - Reference Documents

- In 3.4.2, exception priority was changed (for compliance with ExternalDebugSpec1.0.0).
  - In 3.7.2.2.1, conditions for clearing MPRV in CSR mstatus were changed (for compliance with ExternalDebugSpec1.0.0).
  - In 3.7.2.8.2, conditions for clearing action in CSR tdata1 were changed (for compliance with ExternalDebugSpec1.0.0).
  - In 3.7.2.8.2, the description of maskmax in CSR tdata1 was changed to one in which match 1 was not supported.
  - In 3.7.2.8.2, SC/AMO(Write) was added as a target instruction for store in CSR tdata1.
  - In 3.7.2.8.2, LR/AMO(Read) was added as a target instruction for load in CSR tdata1.
  - 9.1
  - Appendix C
- Elimination of configuration parameter restrictions about AXI configuration
  - 3.8.2.1
  - 3.9.1
  - 10.2.1
  - 12.1
- OLD PLIC specification document was deleted from Reference Documents.
- In 1.1 Features, interrupt acknowledgment unit was changed from ICU to ACLINT/PLIC.
- In 1.2, ICU was deleted from Functional Block Diagram.
- In 2.1.1, Address Region List was updated.
- In 2.1.1.4, Internal Register Region Address Map was changed.
- In 2.1.1.8, Hart Context Register Region Address Map was changed.
- In 2.1.1.9, Debug Subsystem Register Region Address Map was changed.
- In 2.1.2, Space That Can Be Accessed from Each Master (AXI configuration) was changed.
- In 2.1.2, the description about accessible spaces was changed.
- In 2.3.2.1, all instruction cache-related registers were changed to memory mapped registers.
- In 2.3.3.1, cache enable/disable procedures were changed.
- In 2.3.3.2, cache clear procedure was changed.
- In 3.3.1, DBG bus master function was deleted from those suspended in Master Stop state.
- In 3.4, exception code was changed to support Resumable NMI.
- In 3.4, exception detail factor codes were deleted.
- In 3.4.1, unique exception list was changed to support Resumable NMI.

- In 3.4.2, exception priority was changed to support Resumable NMI.
- In 3.5.6, the detailed description about NMI was changed to support Resumable NMI.
- In 3.7.1, Exception Cause Flags was deleted from NS31A CSR Address Map.
- In 3.7.1, Hart/Cache Control was deleted from NS31A CSR Address Map.
- In 3.7.1, Hart Program Counter was deleted from NS31A CSR Address Map.
- In 3.7.2, Exception Error Address was deleted from NS31A CSR Listing.
- In 3.7.2, Hart control was deleted from NS31A CSR Listing.
- In 3.7.2, L1 Instruction Cache control was deleted from NS31A CSR Listing.
- In 3.7.2, Hart Fetch address Register was deleted from NS31A CSR Listing.
- In 3.7.2, mnscratch was added to CSR Listing.
- In 3.7.2, mnepc was added to CSR Listing.
- In 3.7.2, mncause was added to CSR Listing.
- In 3.7.2, mnstatus was added to CSR Listing.
- In 3.7.2, menvcfg was added to CSR Listing.
- In 3.7.2, menvcfgh was added to CSR Listing.
- In 3.7.2, mstatush was added to CSR Listing.
- In 3.7.2, mconfigptr was added to CSR Listing.
- In 3.7.2.1.3, SIF field was deleted from fcsr.
- In 3.7.2.2.2, a description about CSR misa bit1 was changed.
- In 3.7.2.5.3, a description about CSR mcause bit30-28 was changed.
- In 3.7.2.2.6, a description about CSR mstatush was added.
- In 3.7.2.3, Machine Configuration CSR category was added.
- In 3.7.2.3.1, a description about CSR menvcfg was added.
- In 3.7.2.3.2, a description about CSR menvcfgh was added.
- In 3.7.2.6, Machine Non-Maskable Interrupt Handling category was added.
- In 3.7.2.6.1, a description about CSR mnscratch was added.
- In 3.7.2.6.2, a description about CSR mnepc was added.
- In 3.7.2.6.3, a description about CSR mncause was added.
- In 3.7.2.6.4, a description about CSR mnstatus was added.
- In 3.7.2.8, Exception Error Address was deleted from CSR description.
- In 3.7.2.10, Hart control was deleted from CSR description.
- In 3.7.2.11, L1 Instruction Cache control was deleted from CSR description.

- In 3.7.2.11.2, hat can be read with time was changed to ACLINT timer counter value.
- In 3.7.2.11.6, what can be read with timeh was changed to ACLINT timer counter value.
- In 3.7.2.12.3, mimpid value was changed.
- In 3.7.2.12.5, a description about CSR mconfigptr was added.
- In 3.7.2.14, Hart Fetch address Register was deleted from CSR description.
- In 3.8, Hart context register specifications were changed.
- In 3.8.2, Hart Context Register bus error occurrence factor was changed.
- In 3.8.2.1, MMIO registers of RISC-V CSR were deleted from Hart context register List.
- In 3.8.3, a description about Hart Context Register was added to Register descriptions.
- In 3.8.3.2.1, meecaddr was moved to Memory Mapped Registers.
- In 3.8.3.2.2, meecunlock was moved to Memory Mapped Registers.
- In 3.8.3.2.3, mhtpc was moved to Memory Mapped Registers.
- In 3.8.3.2.4, mhtenable was moved to Memory Mapped Registers.
- In 3.8.3.2.5, mhtstatus was moved to Memory Mapped Registers.
- In 3.8.3.2.6, ml1cachesysctrl was moved to Memory Mapped Registers.
- In 3.8.3.2.7, ml1cacheoperation was moved to Memory Mapped Registers.
- In 3.8.3.2.8, ml1cacheprefcfg was moved to Memory Mapped Registers.
- In 3.8.3.2.9, mbpsysctrl was moved to Memory Mapped Registers.
- In 3.8.3.2.10, mbpoperation was moved to Memory Mapped Registers.
- In 3.8.3.2.11, mhtfetchaddr was moved to Memory Mapped Registers.
- In 3.8.3.14, GPR0 MMIO register was deleted.
- In 3.8.3.15, GPRn MMIO register was deleted.
- In 3.8.3.16, FPRn MMIO register was deleted.
- In 3.9, RV32B was added to the supported instruction sets.
- In 3.9.6.1, mnemonic of unique extended instructions was changed to ns.fexp2.s and ns.flog2.s.
- In 3.9.6.1, a description about subnormal number output for unique extended instructions was added.
- In 3.9.6.1, opcode in unique extended instruction encoding was changed to custom-1.
- In 3.9.6.2, range of flushing of subnormal numbers was changed to include only unique instructions.
- In 3.9.7, NS31A Bit Manipulation extension was changed to RV32B extension.
- In 3.9.7.1, unique instructions were changed to the instructions supported by RV32B.
- In 3.10.2.1, reservation release conditions of LR instructions were changed.
- In 3.12.3, behavior summary was changed according to changes in rnmi specifications.

- In 4.1.1, DBG-initiated access was deleted.
- In 6.2.2.10, FUNCCTRL.FFCSRSIF field was disabled.
- In 6.2.2.23, MIMPID value was changed.
- In 9.3.2.2.2, value of Version in IDCODE was changed.
- In 11.1, mnemonic of unique extended instructions was changed to ns.fexp2.s and ns.flog2.s.
- In 12.1, default value of P_ADDR_BASE_REG in Configuration Parameter List was changed.
- In 12.1, P_EXTOUT_WIDTH was deleted from Configuration Parameter List.
- In 12.1, default value of P_ADDR_BASE_DLM was changed.
- In 12.1, the description of P_SYNC_STAGES_* was changed.
- In 12.2.1, int_intout pin was deleted from NS31A UIL Port List.
- In 12.2.1, ncc_mtime pin was deleted from NS31A UIL Port List.
- In 12.2.2, int_intout pin was deleted from NS31A CX Port List.
- In 12.2.2, ncc_mtime pin was deleted from NS31A CX Port List.
- In 12.2.2, DBG-TL32N Slave was deleted from NS31A CX Port List.
- In 12.2.2, DBG-Status Control was deleted from NS31A CX Port List.
- In 12.2.2, db31a_status_core_active in DBG-Core Status was deleted from NS31A CX Port List.
- In 12.2.2, DBG-Core Control pin names in NS31A CX Port List were changed.
- In 12.2.2, DBG-APB Master pin names in NS31A CX Port List were changed.
- In 12.2.3, NS31A DBG Port List was changed.
- In Appendix A, MMIO registers of RISC-V CSR were deleted from NS31A CSR Register List.
- In Appendix A, Hart Context Register was deleted from NS31A CSR Register List.
- In Appendix B, base addresses and addresses were changed according to changes to address map specifications.
- In Appendix B, Hart Context Register Region was added to NS31A Internal Register Region.

Supplementary Explanation

- 2.1.2 Explanation of bus access routing were separated between AXI and AHB configurations.
- 2.3.3.4 Section of Diagnosis Access was added.
- 3.3.3 Explanation of Execute ebreak in U-mode was added.
- 3.4.1 Description of unique exception specification was moved from ICU chapter to this section.
- In 3.4.1, the description about exception factor codes were moved to this section.
- In 3.8.2, the description about Debug Mode Category registers was deleted because Hart Context Registers no longer contains such registers.

- In 3.9.6, the description about unique additional instructions was moved forward because the specifications for flushing of subnormal numbers were limited.
- 3.10.1 Explanation of AMO operations was added.
- 3.10.2 Explanation of LR/SC operations was added.
- In 4.1, the description about Read Data Interleaving was added.
- 11.1 Description was changed in P_M_AHB
- Acknowledge sequences of multiple interrupt were changed.
    - 7.3.3.3.2
    - 7.3.3.4.2

Differences between Revision 1.0.1 and Revision 2.0.0

Additional Specifications

- AMBA AHB was added to Reference Documents.
- In the function overview, descriptions about additional functions were added.
- The specifications for the AHB configuration were added.
    - In 1.2, block diagrams were added.
    - 2.1.1
    - 2.1.1.1
    - 2.1.1.2
    - 2.1.1.3
    - 2.1.1.4
    - 2.1.1.5
    - 2.1.1.6
    - 2.1.2
    - 2.2
    - 3.8.1.1
    - 3.8.2
    - 3.10.1
    - 3.10.2
    - 3.10.2.1
    - 3.10.3
    - 3.10.4

    - 3.12.3
    - 4.1
    - 4.1.1
    - Section 4.1.4 Instruction Bus Master Interface was added.
    - Section 4.1.5 Data Bus Master Interface was added.
    - Section 4.1.6 Peripheral Bus Slave Interface was added.
    - Section 4.2.1.3 Instruction Bus Master Sideband was added.
    - Section 4.2.1.4 Data Bus Master Sideband was added.
    - 9.3
    - 9.6
- The specifications applicable when parameter P_CACHE_SIZE_IC is set to 0 was added.
    - 2.2
    - 2.2.1.1
    - 2.3
    - 2.3.1
    - 2.3.3.2
    - 2.3.3.3
    - 3.7.2.11.1
    - 3.7.2.11.2
    - 3.7.2.11.3
    - 3.8.3.7
    - 3.8.3.8
    - 3.8.3.9
    - 3.8.3.10
    - 3.12.2.1
    - 3.12.4
    - 6.1.4
    - In 6.2, the specifications for cache clear-related registers and system control registers in the AHB configuration were added.
        - Section 6.2.2.8 Cache Clear Status Register (STATCCLR) was added.
        - Section 6.2.2.9 Cache Clear Control Register (CTRLCCLR) was added.
        - 6.2.2.15

- RV32C was added to the instruction sets.
  - 3.4.2
  - 3.7.2.2.2
  - 3.7.2.4.2
  - 3.7.2.7.2
  - 3.7.2.10.1
  - 3.9
  - Section 3.9.8. Compressed Instructions (RV32C) was added.
  - 3.10.2.1 Table 102.
- The specifications applicable when changing the P_BTB parameter were added.
  - 3.1.1
  - 3.7.2.11.4
  - 3.7.2.11.5
  - 3.8.3.11
  - 3.8.3.12
  - 3.8.3.13
  - 3.11
  - 3.12.4
- The specifications applicable when changing the P_PMPNUM parameter were added.
  - 3.2.1
  - 3.7.2.5.1
  - 3.7.2.5.2
  - 9.1
  - 9.2
  - 9.2.1
- RV32E was added to the instruction sets.
  - 3.7.1
  - 3.7.2.2.2
  - 3.8.3.15
  - 3.9
  - 3.9.1
- Information to indicate that access might cause a bus error depending on the parameter specifications was added.

  - 3.8.1.1
  - 3.8.2.1
- The specifications about the system-level states in the AHB configuration were added.
  - 3.3.1
  - 5.3.1
  - 5.3.2
  - 5.3.3
- The configuration parameters for an integer multiplier and divider were added.
  - Section 3.12.7 Optimization of the integer multiplier and divider was added.
- The CMU configuration parameter registers were added.
  - 6.1.5
  - 6.2.1
  - Section 6.2.2.20 PMP Entry Number Configuration Register (CFGPMPNUM) was added
  - Section 6.2.2.21 Instruction Cache Size Configuration Register (CFGCSIZEIC) was added
- Information about the System Bus Access function deletion of DBG in the AHB configuration was added.
  - 8.1.3.6
  - 8.1.6.1
  - 8.2.1
  - 8.2.2.1
  - 8.2.2.2.13
  - 8.2.2.2.14
  - 8.2.2.2.15
  - 8.2.3.4
- The function safety specifications applicable when RAM was not mounted were added.
  - 9.1
  - 9.5
- In 10.1, cfg_resetvec was changed to be the default value of the PC when the reset is released.
- In 11.1, the configuration parameters were added and changed in line with function extension and deletion.
- Pin lists for the AHB configuration were added
  - Section 11.2.5 NS31A UIL Port List (AHB Configuration) was added
  - Section 11.2.6 NS31A CX Port List (AHB Configuration) was added
  - Section 11.2.7 NS31A DBG Port List (AHB Configuration) was added

  - Section 11.2.8 NS31A MEM Port List (AHB Configuration) was added

Specification Change

- The specifications about hart IDs were changed (fixed to 0 ⇒ support 0 to 31).
  - 3.7.2.13.4
  - 8.1.4.2
- In 3.7.2, "RV32I only" was changed to "RV32 only"in Table 20.
- In 3.7.2.2.2, the value of bit 1 of the misa register was changed.
- In 3.7.2.10.1, the conditions for saving the mhtpc pc value and recovering the pc value were changed.
- In 3.7.2.10.2, the initial value of the mhtenable register was changed.
- In 3.7.2.10.3, the initial value of the mhtstatus register was changed.
- In 3.8.2.1, the register access behavior expected when changing the number of PMP entries in the AXI configuration was changed.
- The bus response in the AXI configuration was modified.
  - 3.8.2.1
  - 11.1
- In Section 4.1.6 Peripheral Bus Slave Interface, the specification of PPROT[2] was changed.
- In 7.3.3.3.2, the procedure for acknowledging the edge detection type interrupt (with multiple interrupts) was changed.
- In 7.3.3.4.2, the procedure for acknowledging the level detection type interrupt (with multiple interrupts) was changed.
- In 8.2.3.1, the halting procedure was changed.
- In 8.2.3.3, the block transfer procedure using the abstruct command was changed.
- In 11.2.1, the jtag_tck synchronous signal was deleted and the JTAG pin synchronous signal was changed to jtag_tck in Table 220.
- The ID codes were updated.
  - 3.7.2.13.3
  - 6.2.2.23
  - 8.3.2.2.2

Supplementary Explanation

- In 2.1.2, a note about the space that can be accessed from each master was added.
- In 2.3.3.2, information was added to the cache clear procedure using the CSR register.

- Mistakes from previous products were corrected.
    - 2.2.1
    - 2.2.1.1
- Supplementary explanations in line with the addition of the AHB configuration were added.
    - 1.1
    - 3.1
    - 3.10.2
    - 8.3
- The descriptions were clarified
    - 3.2.1
    - 3.4
    - 3.7.2.6.1
    - 3.7.2.6.4
    - 3.7.2.7.1
    - 3.7.2.7.2
    - 3.7.2.7.3
    - 3.9.6.3
    - 3.11
    - 8.1.4.4
- The order of 3.7.2.10.2 mhtstatus and 3.7.2.10.3 mhtenable was changed according to the order of addresses.
    - 3.7.2.10.3
    - 3.7.2.10.2
- In 3.8.3.16, the specifications for changing the P_FPU parameter were added.
- In 4.1.1, information about the transaction ID bit width was added.
- In 4.1.3, information about the transaction ID bit width was added.

Differences between Revision 1.0.0 and Revision 1.0.1

Additional Specifications

- Descriptions about the illegal instruction exception conditions for accessing the User Floating-Point CSRs (fflags, frm, fcsr) were added.
- In 3.7.2.10.1, a description about the behavior when a Resume request is issued while the Hart-level state is DB-Halt and the value of PEnable in mhtenable is 0 was added.

- In 3.7.2.10.1, a description about the behavior when a Halt request is issued while the Hart-level state is Disable or Stop was added.
- In 3.7.2.10.1, the points to be noted when the hart is resumed from DB-Halt to Stop or Disable were deleted.
- Section 5.2.3 JTAG Reset was added.

Specification Change

- In Table 13, the value to be stored in mtval for Reset (mcause=0000_000h) was modified from 0h to Unknown.
- In Table 13, the value to be stored in mstatus.mpp for Reset (mcause=0000_000h) was modified from 3 to 0.
- In 3.4, the value to be stored in mtval for {Software, Timer, External} interrupt was modified from s to 0h.
- In Table 14. mtval value for instruction fetch, the detail cause when mtval=000000A0h was corrected.
- In 3.7.2.13.3, the value of mimpid was changed.
- In 3.8.3.8, the INDEX field bit width of the L1 instruction cache self-diagnosis access register (L1CRATGT) was corrected from [8:0] to [10:0].
- In 3.8.3, the description about the data to be stored in the L1CRADAT0 and L1CRADAT0 registers was modified to add the descriptions for INDEX = 4 to 7.
- In 3.10.2, the specifications of the memory regions to which the LR/SC instructions cannot be executed were modified.
- In 3.12.4, the behavior for the L1 instruction cache if the FENCE.I instruction is executed was modified from “not cleared” to “cleared”.
- In 5.2, the restrictions on controlling rst_pon and jtag_trst_n were added.
- In 6.2.2.21, the value of MIMPID was changed.

Supplementary Explanation

- In 3.7.2.10.1, the points to be noted when the hart is resumed from DB-Halt to Stop or Disable were added.
- In 3.10.4.2, the point to be noted about the condition for canceling the exclusive DLM monitor reservation was added.
- In 3.12.4, the point to be noted when an instruction code on the memory is rewritten to the EBREAK instruction in Debug mode was added.

NSI-TEXE

■Contact

For sales and technical inquiries, contact us using a contact form provided in our website.
URL: https://www.nsitexe.com/en