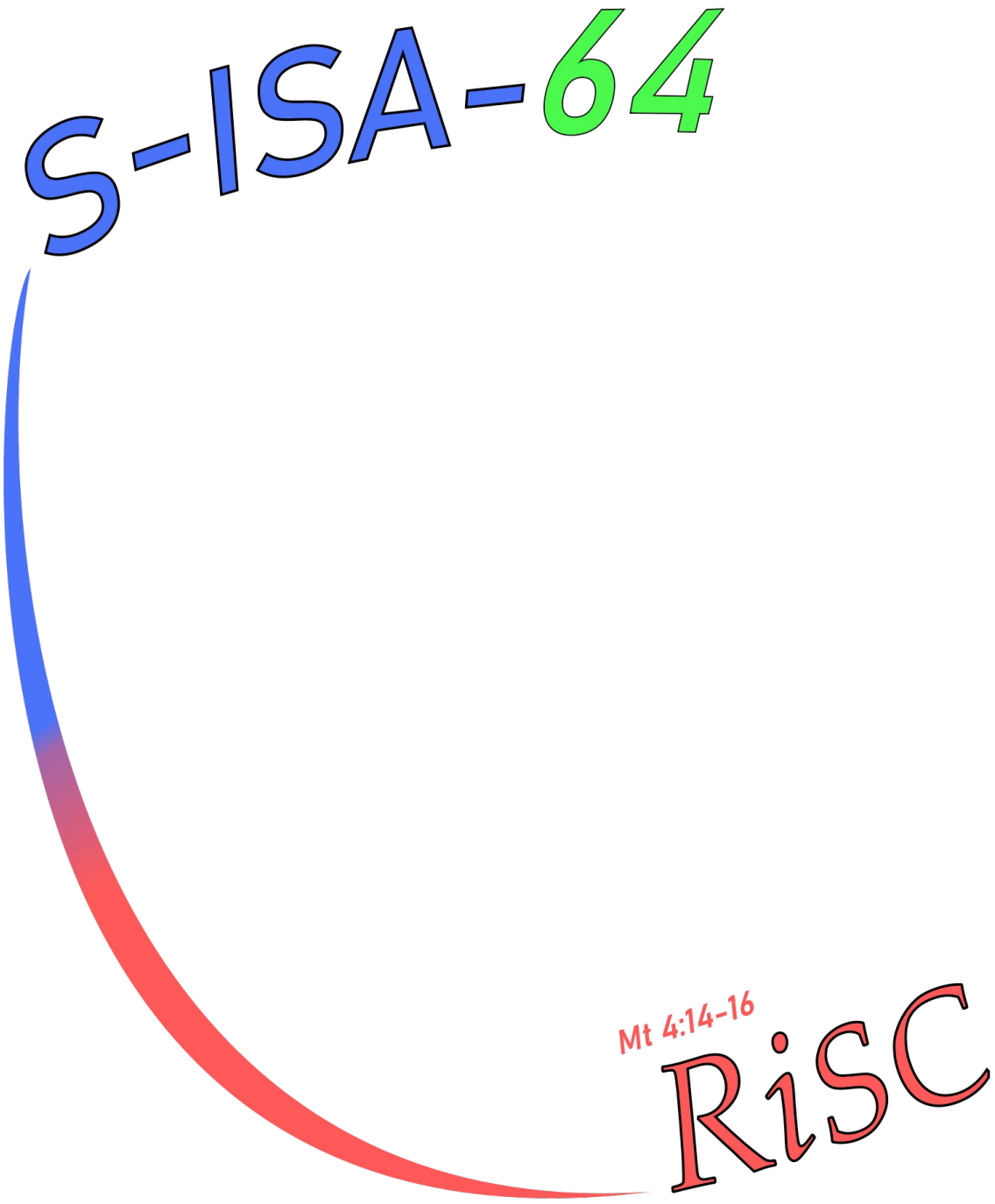
S-ISA-64 Device Standard



Contents

Overview....3
Prerequisite Terminology and Concepts....4
Device Interaction- Execution Flow....5
Default Device Behavior....6

Overview

S-ISA-64 accesses non-RAM components through two instructions: the `dread` and `dwrite` instructions, which send data and commands on a special bus dedicated for interacting with peripheral components.

The Programmer's Manual covers the overall functionality of these two instructions in the instruction set, while this document aims to document how components on the system (such as non-volatile storage, a graphics card, keyboard, display, microphone, ethernet controller, or USB hub). Are discovered and interacted with in terms of `dread` and `dwrite`.

It also documents the errata of multi-threaded access to the device bus and what interactions must be implemented.

The Programmer's Manual is considered a *Necessary Companion* for understanding this document.

Prerequisite Terminology and Concepts

This is what this whole document is actually about:

`dread`- instruction in the S-ISA-64 instruction set which **read**s 64 bits from a **d**evice on the **d**evice bus- a special address space which behaves like memory that can be read or written to. The actual instruction takes a single register ID operand and writes the return value to the same, but the effective functionality is to take in an address and return a 64 bit integer. Thus, the prototype is

```
uint64 dread(uint64 address);
```

`dwrite`- instruction in the S-ISA-64 instruction set which **write**s 64 bits to a **d**evice on the **d**evice bus. The actual instruction takes two register ID's as operands, but the effective functionality is to pass in an address and a value, and return nothing. Thus, the prototype is:

```
void dwrite(uint64 address, uint64 val);
```

These two operations form the basis of all interactions with components (excluding main memory) on the system. Components on the system may also be configured (through dwrite operations) to take data from main memory, if supported. (How devices with such functionality behave and how they are configured shall be documented in this standard.)

This document's primary goal is to create a hardware-software interaction standard for the S-ISA-64 architecture **before** hundreds of hours of work has been spent to design the actual hardware, the emulator, operating system development kit, and system libraries.

Unless otherwise specified, Anything called an "address" in this document is an address on the *device bus* and not a main-memory address.

Device Interaction- Execution Flow

Upon boot, the operating system on the computer will check its own memory size by reading address 1, check for the presence of standards-compliant coprocessor cores in address 3, and mutual exclusion devices (“mutexes”) in address 5. Using these values, if applicable, it may read the pointer to the *processor kickstart addresses* in address 4. After reading 1 and 4, the operating system may also try to use one of the *processor kickstart addresses* to start a secondary core. How processor kickstart addresses work shall be documented later. It may also read the pointer to the *Mutual Exclusion Device Addresses* in Address 6. It may also try to retrieve the starting address of Processor Status Variables from Address 7.

Here, we have our first standard rules:

1. ***The memory size and millisecond clock functionalities of addresses 1 and 2 are considered vital to a system and shall not be absent or mangled in a compliant implementation.***
2. ***Any default device addresses which are not implemented as according to this standard, shall safely return zero, or do nothing on a write.***

Finally, and most importantly, the operating system may check for attached devices by checking address 8, and if the return value is non-zero it may check the individual address spaces of the available peripherals

3. ***If there are no secondary peripherals on the system, address 8 returns 0.***
4. ***If the return value of address 8 is non-zero, then reading the first addresses in the address spaces of the corresponding peripherals must be defined behavior.***
5. **(TODO) define how plug and play devices work.**

Default Device Behavior

The default device, Peripheral 0, is the heart of the implementation and is vital to the functionality of any operating system.

Serial I/O UART (Address 0)

The Serial I/O device provides blocking read/write of ASCII character values for debugging purposes. It is not considered essential for a S-ISA-64 implementation as its primary purpose is debugging, but this address must only be used for that purpose. When dread is invoked on the address, the processor is halted until a value is retrieved from the input device. An 8-bit ASCII character (the bottom 7 being the 7-bit ASCII standard) value is returned in the bottom 8 bits of the 64 bit return value. When dwrite is invoked on the address, the bottom 8 bits of the input are treated as an 8-bit ASCII value, and should be displayed, transmitted, or broadcast as such to the appropriate serial port.

Memory Size (Address 1)

The Memory Size query device returns the amount of available system RAM for the implementation. Writing to this address is undefined behavior, but reading it shall return the number of available addressable contiguous bytes to the S-ISA-64 processor starting at address zero.

Clock (Address 2)

The Clock device returns a measure of the number of milliseconds passed since some point in the past to the caller on a dread. It must provide at least 32 bits of wrap-around (It may wrap around every 4,294,967.296 seconds).

Processor Count (Address 3)

The processor count device returns the number of available compliant processors on the system, which is dubbed **Np**. If the implementation is single-threaded, it may return either 0 (unimplemented) or 1. The behavior of Processors shall be documented in *Multiprocessor Essentials.*

Processor Kickstart Address Array Pointer (Address 4)

Reading address 4 shall return a valid device bus address (Dubbed "**Kickstart[0]**"), which, if written to, attempts to start the first processor on the system (Which is the ***boot processor***). The next address ("**Kickstart[1]**") attempts to start the second processor on the system, if available, and so on, up to **Kickstart[Np-1]** where **Np** is the number of compliant processors on the system, as provided by address 3.

If there are no available kickstart addresses, or they are unimplemented, Address 4 shall return 0.

If reading address 3 returns 0, then reading address 4 must also return 0.

Mutex Count (Address 5)

Address 5 shall return the number of available contiguous mutual exclusion devices on the system, which is dubbed **Nm**. If none are available, zero is returned.

Mutual Exclusion Device Array Pointer (Address 6)

Address 6 shall return a valid device bus address (Dubbed "**Mutex[0]**"), which, if a value of 0 is written to it, shall attempt to "unlock" it. And if a value of 1 is written to it, shall attempt to "lock" it. Writing other values is *implementation defined behavior.* The next address, ("**Mutex[1]**") shall behave identically to Mutex[0] if **Nm** is at least 2.

If reading address 5 returns 0, then address 6 must also return 0.

Processor Status Variable Array Pointer (Address 7)

Reading Address 7 shall return a valid device bus address (Dubbed "**Pstat[0]**") which, if read, shall return a value indicating whether or not the processor is running and/or currently unavailable to be kickstarted by writing to its corresponding Kickstart address (which in this case would be Kickstart[0])

If the processor is running, and not able to be kickstarted, the value returned shall be 1.
if the processor is available to be kickstarted, the value returned shall be 0.
if the processor is for any reason unable to be kickstarted, the value returned shall be 0.

The next address, **Pstat[1]** shall behave identically to **Pstat[0]** if available.

The number of processor status variables, if implemented, must be at least as large as the number of processors, **Np**.

If the return value of Address 3 is zero, the return value of Address 7 must also be zero.

Device Descriptor Count (Address 8)
Address 8 shall yield the number of secondary devices (Dubbed "**NeX**") on the system (Dubbed "**Peripherals**"). Each secondary device (Peripherals 1 – **NeX**) has a series of attributes stored at the beginning of its address space.

Pointer to Main Memory Mirror (Address 9)
Returns Device Bus Address where a 64-bit-aligned mirror of main memory can be accessed. This exists to allow DMA controllers to read and write on the device address bus rather than using main memory.

Set Is Serial NonBlocking (Address 10)
Reading this will reveal whether Serial is nonblocking or not (0 or 1)
Writing 0 will make it blocking, and 1 will make it non-blocking.

Set System Time (Address 11)
Reading this will reveal whether the system time can be set (0 or 1).
Writing will set the system time in milliseconds to the value written.

Default Device Vendor String Pointer (Address 12)
Returns a Device Bus Address where a null-terminated ASCII string specifying vendor-specific information (vendor name, model, extensions, whatever) can be read. Vendor strings are read 8 bits at a time, so contiguous addresses yield increasing bytes.
If the vendor string started at 0x100 and the string was "Hi!" then reads would look like this:

```
read(0x100) → 'H'
read(0x101) → 'i'
read(0x102) → '!'
read(0x103) → 0
```

This model of how a "vendor string" works shall be mimicked by all compliant secondary devices.

Secondary Device Standard Layout

Peripherals on a system are identified by a peripheral ID, from 0 (the default device) to 1,048,575.