EDU-RNG-SYNC: Educational Synchronous Random Number Generator PCI Device

Technical Reference Manual

Virtual Hardware Division

September 2026

Contents

0.1 Document Information

Item	Value
Product Name	EDU-RNG-SYNC
Device ID	0xCAFE
Vendor ID	0x1234
Revision	0x03
Interface	Conventional PCI

Item	Value
Generator Type	Pseudo-random
Operation Mode	Asynchronous

1 Overview

The EDU-RNG-SYNC device is a simple memory-mapped PCI random number generator intended for device driver and operating system education.

The device exposes a minimal programming model consisting of a single random-number register and a single seed register. Reading the random-number register immediately synchronously generates and returns a new pseudo-random value. The design is intentionally simple and aims to provide a first experience of PCI driver development.

2 Features

- Conventional PCI device
- Memory-mapped I/O (MMIO) interface
- 32-bit random value generation
- Programmable generator seed
- Immediate synchronous operation
- No DMA, no interrupts, no internal buffering

3 PCI Configuration

3.1 Device Identification

Field	Value
Vendor ID	0x1234
Device ID	0xCAFE
Revision ID	0x03
Class Code	Other Device
Interrupt Pin	None

4 Memory Map

The device exposes a single MMIO Base Address Register (BAR).

4.1 BAR0

Property	Value
BAR Number	0
Size	0x1000 bytes (4 KiB)
Type	Memory-mapped

4.2 BAR0 Layout

Address Range	Description
0x0000-0x0003	RANDOM register
0x0004-0x0007	SEED register
0x0008-0x0FFF	Reserved

5 Register Summary

Unless otherwise stated, all registers are 32-bit little-endian values and shall be accessed using aligned 4-byte transactions.

Offset	Register	Size	Access
0x000	RANDOM	32 bits	RO
0x004	SEED	32 bits	WO

Access legend: RO = Read-only, WO = Write-only.

6 Register Descriptions

6.1 RANDOM Register

Offset: 0x000
Access: Read-only

Bits	Name	Description
31:0	RANDOM_VALUE	Generated pseudo-random number

Reading this register generates and returns a new pseudo-random 32-bit value. Successive reads return different values.

Example:

```
uint32_t random_value = readl(bar0 + REG_RANDOM);
```

6.1.1 Behaviour

- Each read triggers a new random number generation operation.
- No buffering is performed.
- No status polling is required.
- No interrupt is generated.
- Reads of sizes other than 32 bits return zero.

6.2 6.2 SEED Register

Offset: 0x004
Access: Write-only

Bits	Name	Description
31:0	SEED_VALUE	New generator seed

Writing this register seeds the pseudo-random number generator.

Example:

```
writel(12345, bar0 + REG_SEED);
```

6.2.1 Behaviour

- Writing updates the internal generator seed.
- Future RANDOM register reads use the new state.
- Writes of sizes other than 32 bits are ignored.

7 Programming Model

The device implements a simple request-response model. To produce deterministic random sequences:

1. Write a seed value.
2. Read RANDOM repeatedly.

To obtain implementation-defined random values:

1. Enumerate the device.
2. Map BAR0.
3. Read RANDOM as required.

8 Device Behaviour

8.1 Deterministic Sequences

When a seed is programmed using the SEED register, subsequent RANDOM register accesses generate a deterministic sequence. Software may reproduce identical sequences by reusing the same seed value.

8.2 Power-Up State

The startup state of the random number generator is undefined. Software requiring reproducible behaviour should always explicitly program a seed value before reading RANDOM.

9 Appendix A. Constants

```
#define EDU_RNG_VENDOR_ID     0x1234
#define EDU_RNG_DEVICE_ID     0xcafe
#define EDU_RNG_REVISION      0x03
```

10 Appendix B. Register Offsets

```
#define REG_RANDOM              0x000
#define REG_SEED                0x004
```