



lib_random: Random number generation

Publication Date: 2025/10/10

Document Number: XM-011312-UG v1.3.1

IN THIS DOCUMENT

1	Introduction	2
2	Usage	2
3	Example	2
4	API reference	2
4.1	Pseudo random	3
4.2	Ring oscillator random	3
4.3	Switching random numbers off	4
5	Further Reading	5

1 Introduction

This library provides both hardware and software random number generation.

Hardware based generation uses an asynchronous oscillator in the *xcore* device.

2 Usage

To use the module you need to use **lib_random** in your application *CMakeLists.txt*, for example:

```
set(APP_DEPENDENT_MODULES "lib_random")
```

An application should then the **random.h** header file:

```
#include "random.h"
```

3 Example

An example demonstrating how to generate random values using the **lib_random** library is provided in *examples/app_random*

It shows the two different methods for initialising a random number generator (software or hardware seed), and then also shows how to generate either a single random value or populate an array with random values.

To build and run the example, run the following from an XTC tools terminal:

```
cd examples/app_random
cmake -G "Unix Makefiles" -B build
```

The application binaries can be built using **xmake**:

```
xmake -C build
```

To run the application using the simulator, run the following command:

```
xsim bin/app_random.xe
```

The random data values will be printed in the terminal.

4 API reference

There are two random-number APIs available, one API that creates fast pseudo-random numbers using a linear-feedback-shift register, one that slowly creates random bits. A third API enables you to switch the ring oscillator off.



4.1 Pseudo random

The Pseudo random number generator uses a 32-bit LFSR to generate a pseudo random string of random bits. This has known weaknesses but is exceedingly fast. It comprises the following functions:

random_generator_t **random_create_generator_from_seed**(unsigned seed)

Function that creates a random number generator from a seed.

Parameters

- **seed** – seed for the generator.

Returns

a random number generator.

random_generator_t **random_create_generator_from_hw_seed**(void)

Function that attempts to create a random number generator from a ring-oscillator random value into the seed, using an asynchronous timer. This is based on a 16-bit start value. For better randomness, initialise the random number by calling random_ro_get_bits() 32 times.

Returns

a random number generator.

unsigned **random_get_random_number**(REFERENCE_PARAM(random_generator_t, g))

Function that produces a random number. The number has a cycle of 2^{32} and is produced using a LFSR.

Parameters

- **g** – the used generator to produce the seed.

Returns

a random 32 bit number.

void **random_get_random_bytes**(
REFERENCE_PARAM(random_generator_t, g), uint8_t in_buffer[], size_t
byte_count,
)

4.2 Ring oscillator random

This interface uses the on-chip ring oscillators to create a random bit after some time has elapsed. These bits are notionally true random. The bit rate is limited by a constant **RANDOM_RO_MIN_TIME_FOR_ONE_BIT**. The default value is a safe value that should produce random bits in most circumstances. You can lower it in order to generate more random bits per second at a risk of introducing correlation.

void **random_ro_init**()

Function that initialises the ring-oscillator random number generator. Call this once before **random_ro_get_bit()** is called

int **random_ro_get_bit**()

Function that may produce a random bit using the ring-oscillator.

If a random bit is available, then it returns 0 or 1 at random.

If no random bits are available, then it returns a negative value which is the time in ticks to wait before the next bit is available.



Pre

random_ro_init() must be called before invoking this function.

Returns

Random bit, or the negated time to wait in ticks.

4.3 Switching random numbers off

The random library switches on a ring oscillator on startup. If it is no longer required it can be switched off to save some power.

void **random_ro_uninit**()

Function that stops the ring oscillator, slightly reducing overall power consumption.



5 Further Reading

- ▶ XMOS XTC Tools Installation Guide
<https://xmos.com/xtc-install-guide>
- ▶ XMOS XTC Tools User Guide
<https://www.xmos.com/view/Tools-15-Documentation>
- ▶ XMOS application build and dependency management system; *xcommon-cmake*
<https://www.xmos.com/file/xcommon-cmake-documentation/?version=latest>



Copyright © 2025, All Rights Reserved.

XMOS Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. XMOS Ltd makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.

XMOS, XCORE, VocalFusion and the XMOS logo are registered trademarks of XMOS Ltd. in the United Kingdom and other countries and may not be used without written permission. Company and product names mentioned in this document are the trademarks or registered trademarks of their respective owners.

