



## Hardware random overview



# Hardware random overview

Stable: 12.02.2020 - 16:46 / Revision: 12.02.2020 - 16:44

## Contents

1 Article Purpose .....	2
2 Framework purpose .....	2
3 System overview .....	2
<b>3.1 Component description .....</b>	<b>3</b>
<b>3.2 API description .....</b>	<b>3</b>
4 Configuration .....	4
<b>4.1 Kernel configuration .....</b>	<b>4</b>
<b>4.2 Device tree configuration .....</b>	<b>4</b>
5 How to use the framework .....	4
<b>5.1 How to use from char device .....</b>	<b>4</b>
<b>5.2 How to use from sysfs .....</b>	<b>4</b>
6 How to trace and debug the framework .....	5
7 Source code location .....	5
8 To go further .....	5
9 References .....	5

## 1 Article Purpose

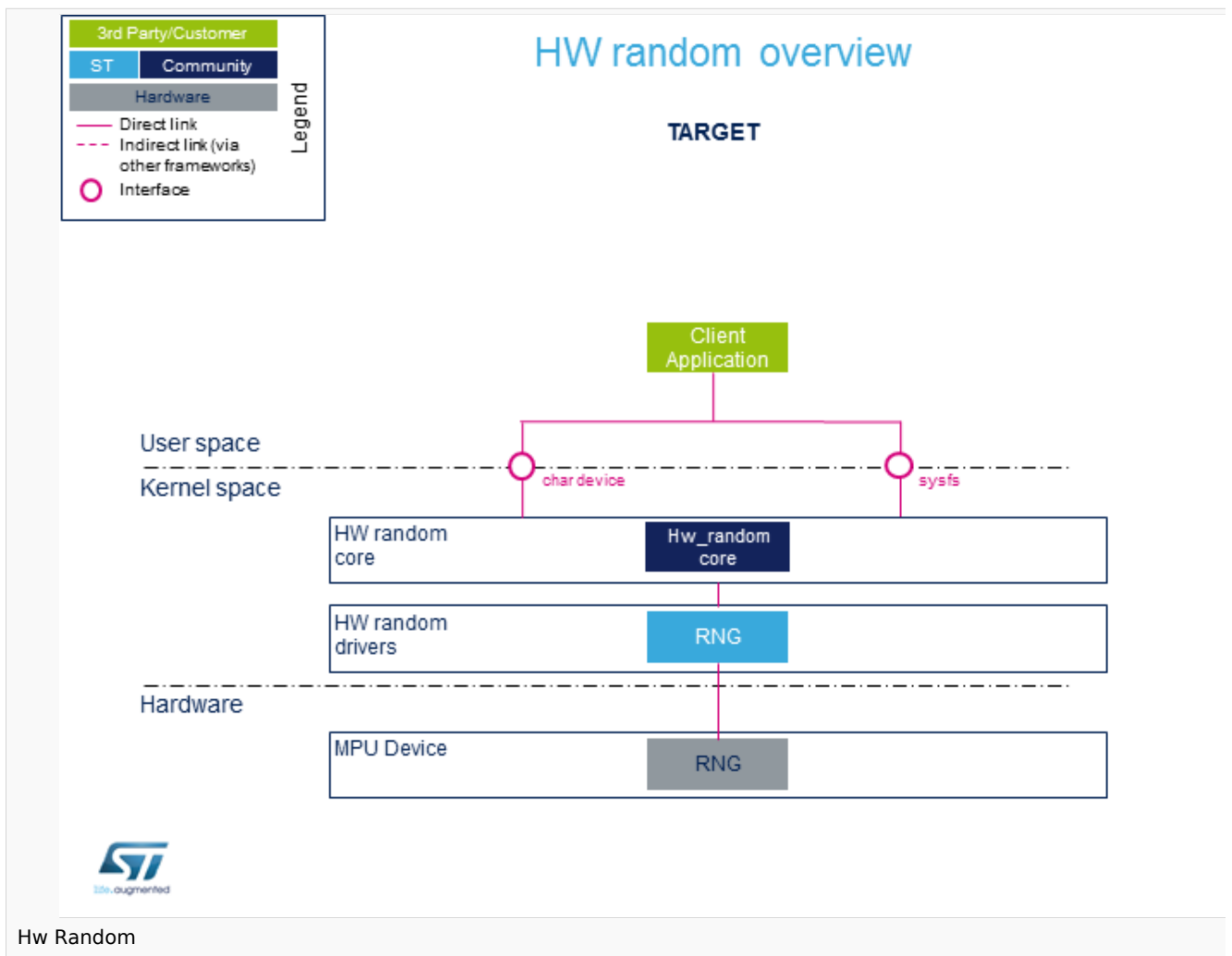
This article gives information about the hardware random (HWRNG) framework.

## 2 Framework purpose

The Hardware random framework is integrated in the kernel. It provides access to RNG peripherals and focuses on supporting the hardware number generator.

## 3 System overview

The HW random framework allows retrieving random numbers in userland.



### 3.1 Component description

- **HW random core** (Kernel space)

Generic interface in kernel space. This layer is in charge of creating the character device (char device) and sysfs to access hw\_random.

- **RNG** (Kernel space)

Hardware random Linux® drivers handling the HW blocks.

- **RNG** (Hardware)

HW blocks handling the RNG peripheral.

### 3.2 API description

The Hardware random framework uses char device API<sup>[1]</sup> ioctl operations. For additional information, refer to:

- sysfs interface.
- Kernel Documentation directory<sup>[2]</sup>

## 4 Configuration

### 4.1 Kernel configuration

The Hardware random support is activated by default in ST deliveries. No specific configuration is required apart from enabling or disabling peripheral support using Linux<sup>®</sup> Menuconfig tool. Refer to [Menuconfig](#) or [how to configure kernel](#) and select:

```
[*] Device Drivers --->
  [*] Character devices --->
    [*] Hardware Random Number Generator Core support --->
      [*] STMicroelectronics STM32 random number generator
```

### 4.2 Device tree configuration

DT configuration can be done thanks to the [STM32CubeMX](#).

A detailed device tree configuration is described in [RNG device tree configuration](#).

## 5 How to use the framework

The framework provides external interfaces from userland : [How to control RNG](#).

### 5.1 How to use from char device

The community tool for using Hardware random framework is [rng\\_tools](#)<sup>[3]</sup> which provides a complete set of utilities related to random number generators:

- **rngd**: runs a background daemon that opens `/dev/hwrng` file (default) to connect and retrieve random numbers.
- **rngtest**: runs different tests that check the entropy and verify the compliance regarding FIPS 140-2 standard.

### 5.2 How to use from sysfs

Available devices compatible with Hardware framework can be listed using `sysfs` commands:

```
Board $> cat /sys/class/misc/hw_random/rng_available
stm32-rng
```

The selected device is shown here:



```
Board $> cat /sys/class/misc/hw_random/rng_current  
stm32-rng
```

To select a different device:

```
Board $> echo "stm32-rng"> /sys/class/misc/hw_random/rng_current
```

## 6 How to trace and debug the framework

---

Light information on the framework can be accessed by using `sysfs`.

By default, the framework does not provide any specific debug output or dynamic debugging tool.

## 7 Source code location

---

Hardware random drivers and framework are available here<sup>[4]</sup>.

## 8 To go further

---

Code examples are directly available from `rng-tools`<sup>[3]</sup> github.

## 9 References

---

- <https://bootlin.com/doc/legacy/accessing-hardware/accessing-hardware.pdf>
- [Documentation/hw\\_random.txt](#)
- [3.03.1 Rng\\_tools source code](#)
- [drivers/char/hw\\_random](#) , [Hw\\_random sources](#)

Random Number Generator

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

Application programming interface

Device Tree