How to control a RNG in userspace

Contents

Stable: 03.02.2020 - 08:42 / Revision: 03.02.2020 - 08:27

Stable: 03.02.2020 - 08:42 | Revision: 03.02.2020 - 08:27

A quality version of this page, approved on *3 February 2020*, was based off this revision.

Contents

1 Purpose

Hardware random framework offers the interface to control RNG devices from userspace.
This article shows two ways to control a RNG in userspace:

- using /dev/random command to generate a random number
- using rng-tools to validate the RNG

2 RNG control through /dev/random

/dev/random is a special file that can be used to generate random numbers based on a pseudo-random generator. It uses noise collected from device drivers and hardware random sources to generate data. od (octal dump) command is used to extract the number of bytes and display the decimal number.
Ex: - Random number (0 - 255):

```
Board $> od -An -N1 -i /dev/random
        172
```

- Random number (0 - 65535):

```
Board $> od -An -N2 -i /dev/random
        20041
```

3 RNG control through rng-tools

rng_tools[1] is a set of tools related to random number generation.
rng-tools will connect to the hardware random number generator through /dev/hwrng. rngtest is a basic test that checks data using FIPS 140-2 tests[2] which is a security requirement test for cryptographic module compliance.

```
Board $> rngtest -c 100 </dev
/hwrng

rngtest 5
Copyright (c) 2004 by Henrique de Moraes Holschuh
This is free software; see the source for copying conditions.  There is NO warranty; not
even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

rngtest: starting FIPS tests...
rngtest: bits received from input: 2000032
rngtest: FIPS 140-2 successes: 100
rngtest: FIPS 140-2 failures: 0
rngtest: FIPS 140-2(2001-10-10) Monobit: 0
rngtest: FIPS 140-2(2001-10-10) Poker: 0
rngtest: FIPS 140-2(2001-10-10) Runs: 0
rngtest: FIPS 140-2(2001-10-10) Long run: 0
rngtest: FIPS 140-2(2001-10-10) Continuous run: 0
rngtest: input channel speed: (min=33.154; avg=33.656; max=34.217) Kibits/s
rngtest: FIPS tests speed: (min=21.193; avg=23.180; max=23.403) Mibits/s
rngtest: Program run time: 58114432 microseconds
```

It is normal for any random generator to fail in small number of tests, but failures must not exceed around 10.

4 References

- https://git.kernel.org/pub/scm/utils/kernel/rng-tools/rng-tools.git/
- https://en.wikipedia.org/wiki/FIPS_140-2

Random Number Generator
Stable: 17.02.2021 - 19:52 / Revision: 17.02.2021 - 19:51

A quality version of this page, approved on *17 February 2021*, was based off this revision.

This article gives information about the Linux® hardware random framework.

Contents

1 Article Purpose

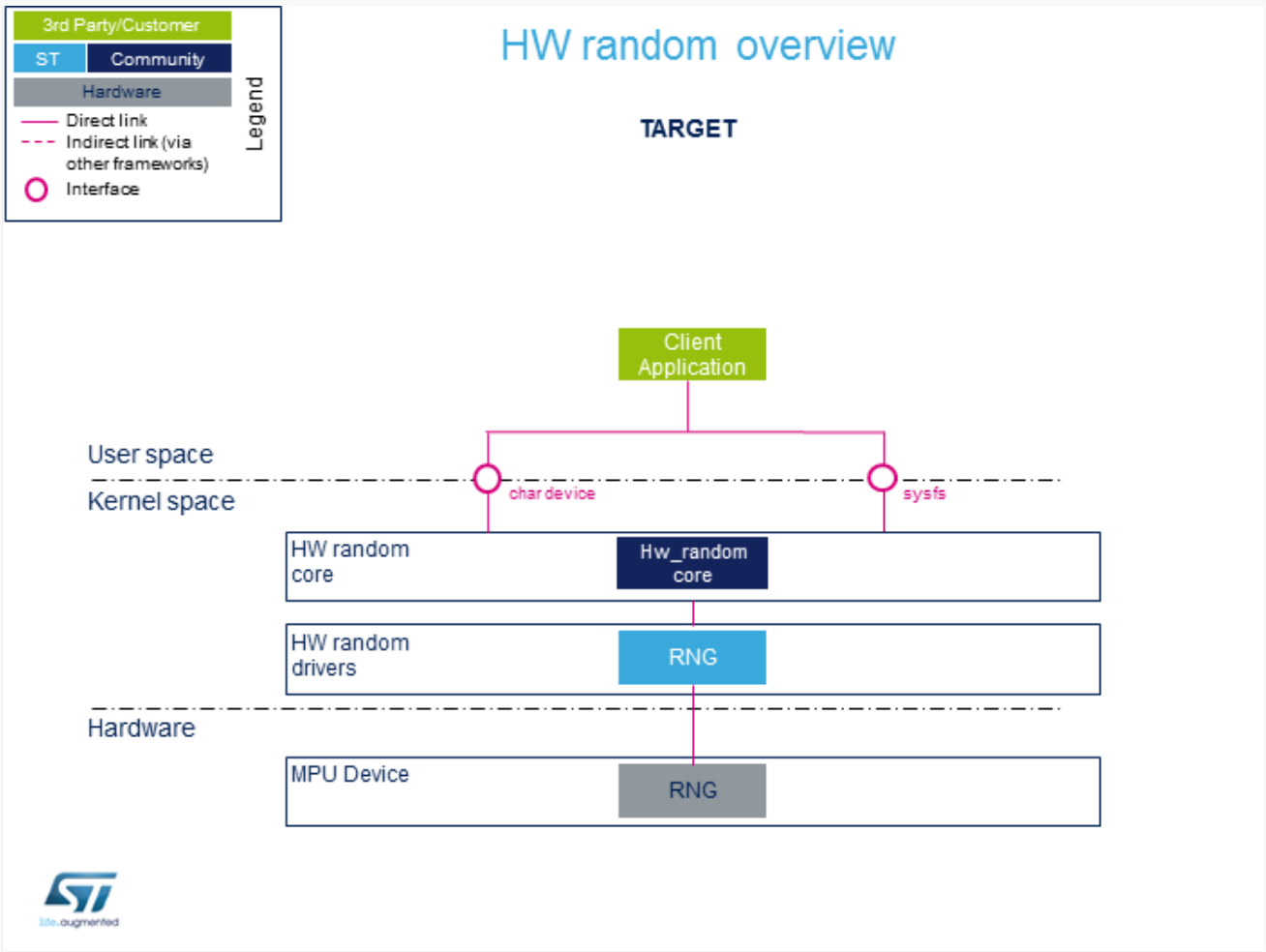
This article gives information about the hardware random (HWRNG) framework.

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.



Hw Random

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[1] ioctl operations. For additional information, refer to:

- sysfs interface.
- Kernel Documentation directory[2]

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® 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[3] 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[4].

8 To go further

Code examples are directly available from rng-tools[3] github.

9 References

- https://bootlin.com/doc/legacy/accessing-hardware/accessing-hardware.pdf
- Documentation/admin-guide/hw_random.rst
- 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