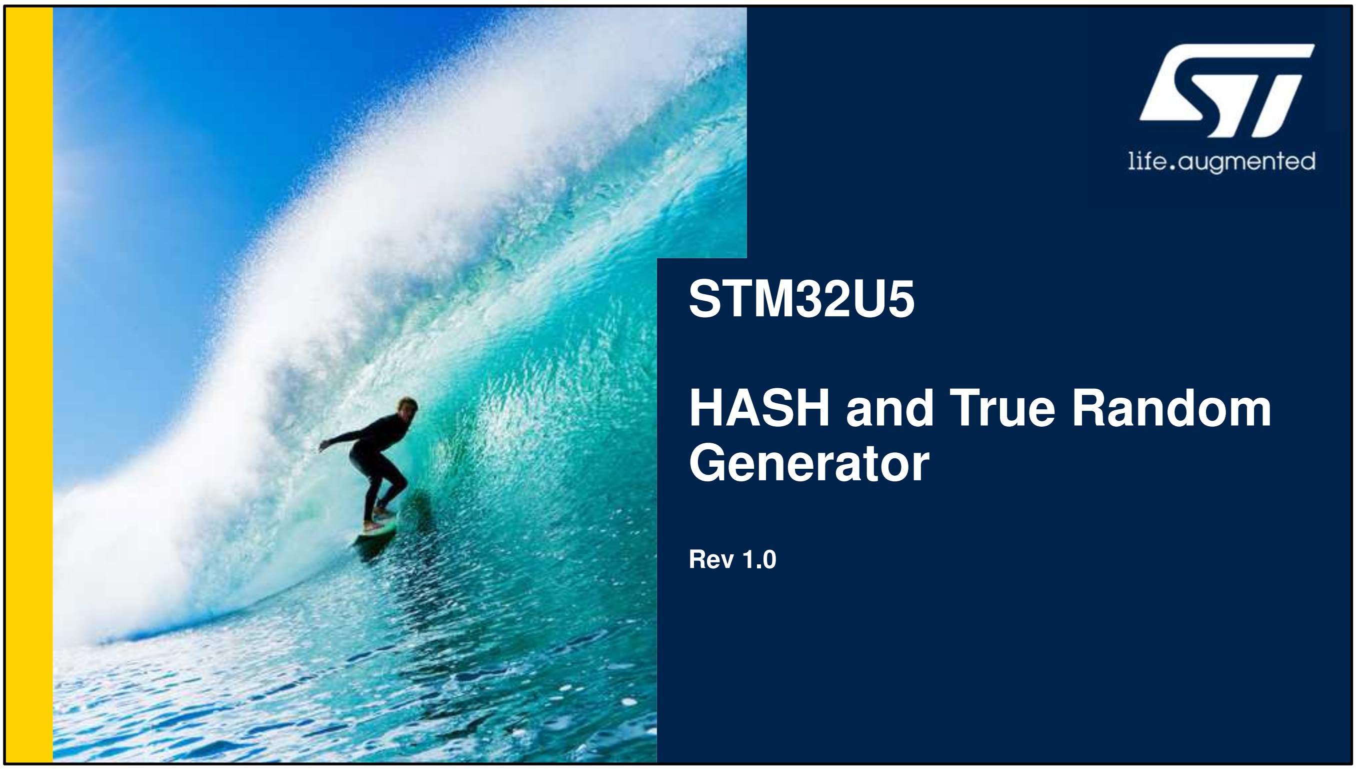
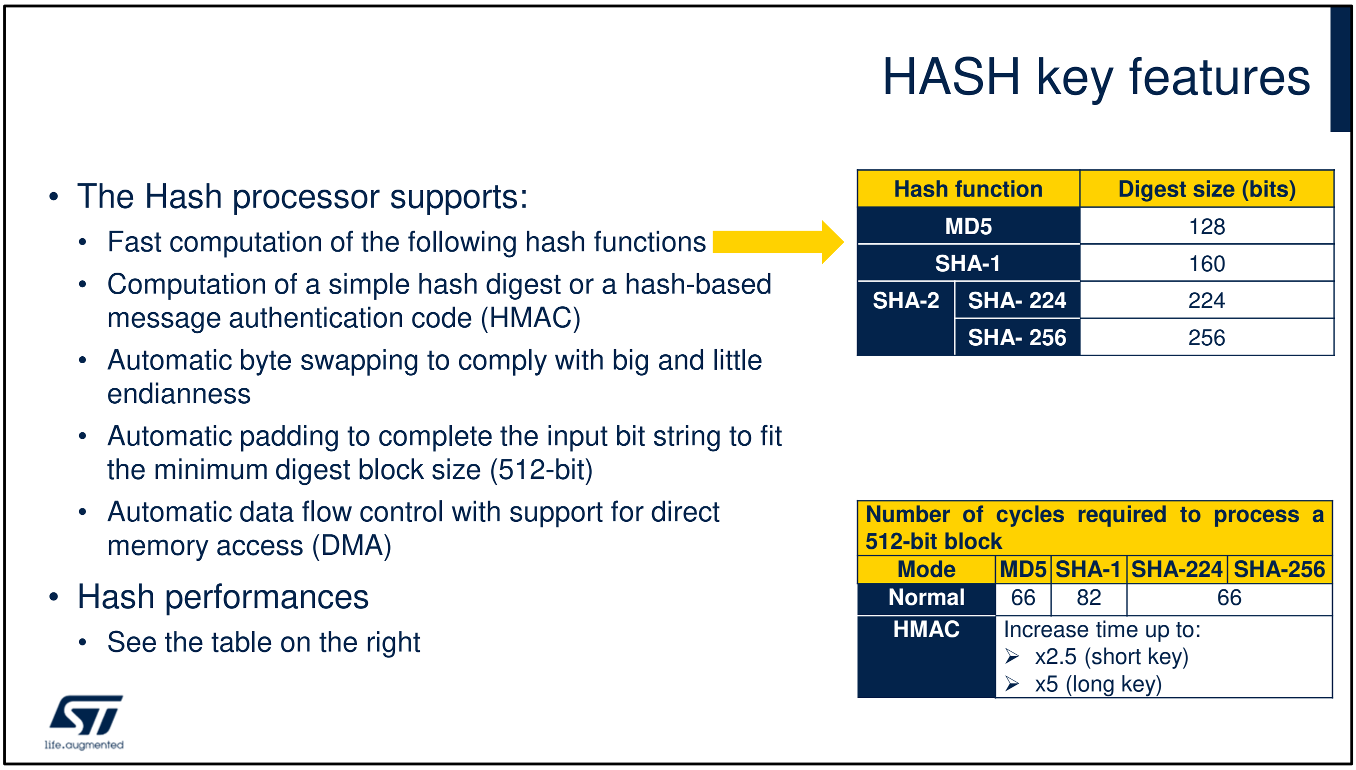


# STM32U5

# HASH and True Random Generator

Rev 1.0

Hello and welcome to this presentation of the S.T.M.32.U.5’s hash processor.



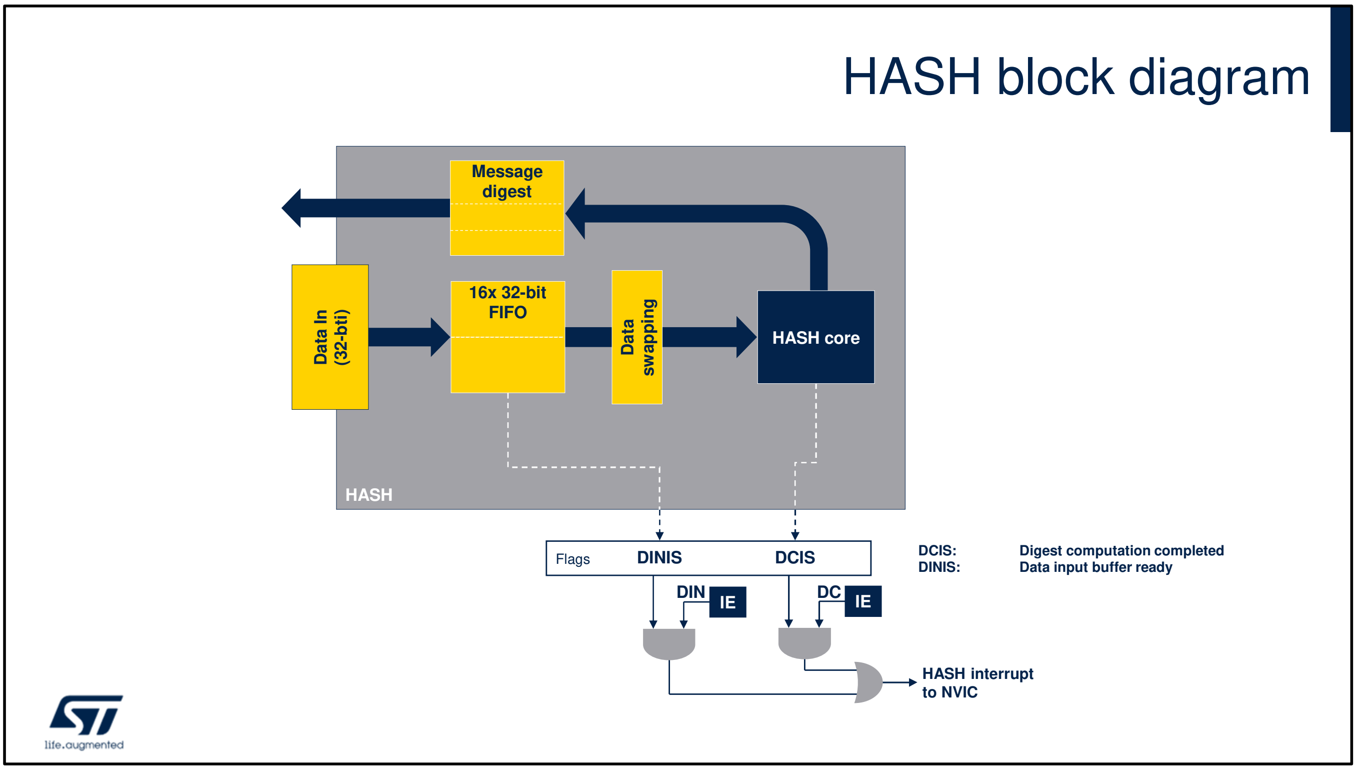
# HASH key features

- The Hash processor supports:
  - Fast computation of the following hash functions
  - Computation of a simple hash digest or a hash-based message authentication code (HMAC)
  - Automatic byte swapping to comply with big and little endianness
  - Automatic padding to complete the input bit string to fit the minimum digest block size (512-bit)
  - Automatic data flow control with support for direct memory access (DMA)
- Hash performances
  - See the table on the right

| Hash function | | Digest size (bits) |
|---|---|---|
| MD5 | | 128 |
| SHA-1 | | 160 |
| SHA-2 | SHA- 224 | 224 |
| | SHA- 256 | 256 |

| Number of cycles required to process a 512-bit block | | | | |
|---|---|---|---|---|
| Mode | MD5 | SHA-1 | SHA-224 | SHA-256 |
| Normal | 66 | 82 | 66 | |
| HMAC | Increase time up to:<br>➢ x2.5 (short key)<br>➢ x5 (long key) | | | |

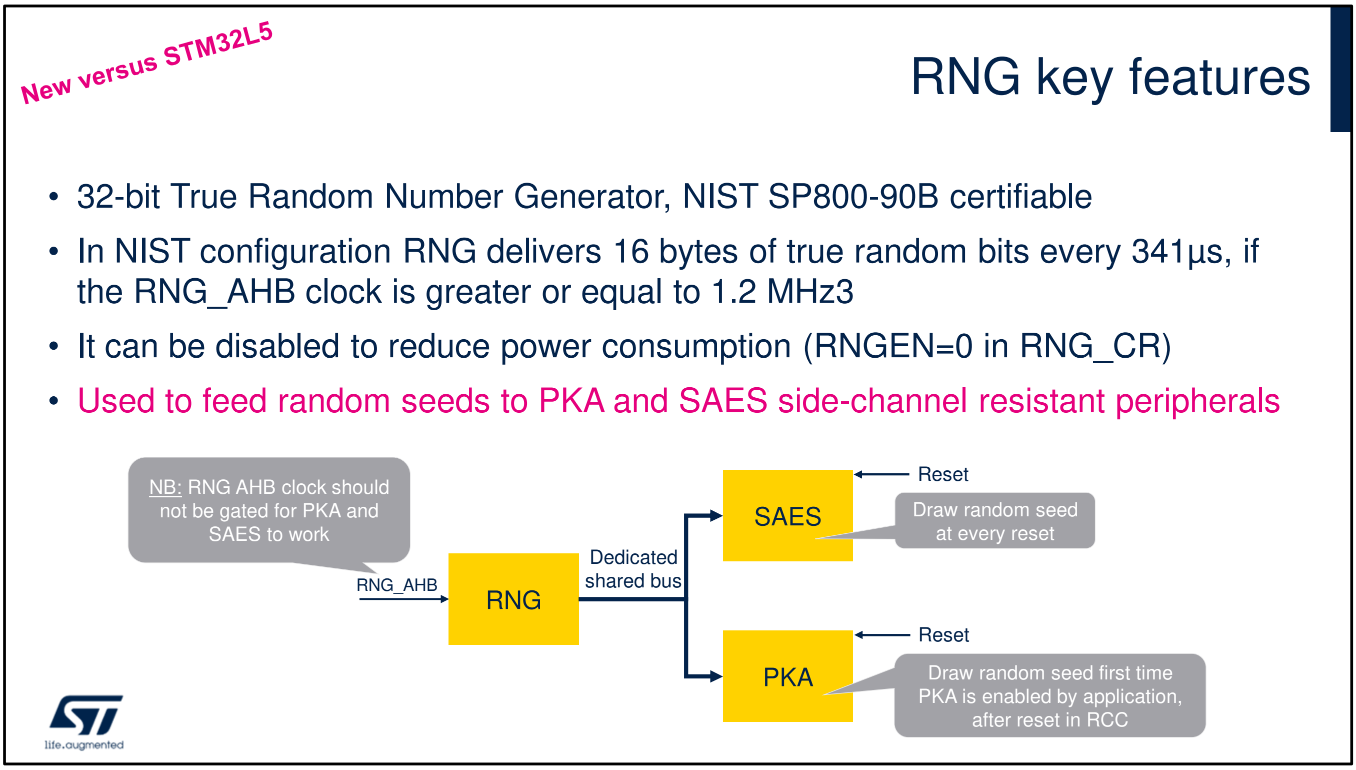
The hash processor supports widely used hash functions including Message Digest 5 (MD5), Secure Hash Algorithm SHA-1 and the more recent SHA-2 with its 224- and 256-bit digest length versions.
When a message of any length is provided as an input, the HASH processing core produces a fixed-length output string called a message digest, depending on the algorithm. The size of the message digest is indicated in the upper table.
A hash can also be generated with a secret-key to produce a message authentication code (MAC).
The processor supports bit, byte and half-word swapping. It also supports automatic padding of input data for block alignment.
The hash processor can be used in conjunction with the DMA for automatic feeding of data.
The times it takes to process a single block of data depends on the chosen algorithms.
The bottom table summarizes the time required to process an intermediate block for each mode of operation.
It is increased by a factor of 2.5 or 5 when an HMAC is also generated.



## HASH block diagram

DCIS: Digest computation completed
DINIS: Data input buffer ready

This simplified block diagram shows that the hash processor processes data blocks through a FIFO and generates digests once the message has been fully loaded.
Input data may be swapped before entering the core unit.
The hash processor manages two individual maskable interrupt sources in the event of a digest calculation completion (DCIS) or a data input buffer ready (DINIS).



New versus STM32L5

# RNG key features

- 32-bit True Random Number Generator, NIST SP800-90B certifiable
- In NIST configuration RNG delivers 16 bytes of true random bits every 341µs, if the RNG_AHB clock is greater or equal to 1.2 MHz3
- It can be disabled to reduce power consumption (RNGEN=0 in RNG_CR)
- Used to feed random seeds to PKA and SAES side-channel resistant peripherals

NB: RNG AHB clock should not be gated for PKA and SAES to work

RNG_AHB

RNG

Dedicated shared bus

SAES

Reset

Draw random seed at every reset

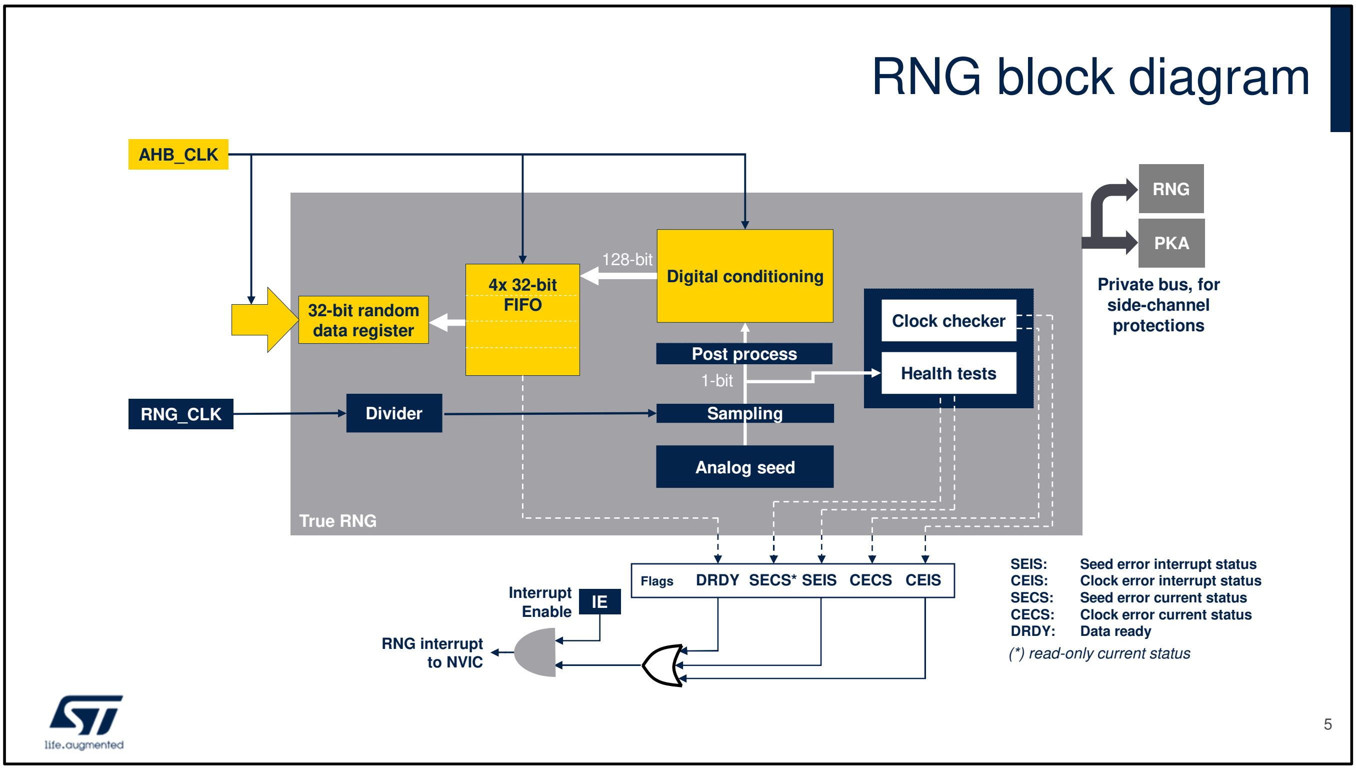
PKA

Reset

Draw random seed first time PKA is enabled by application, after reset in RCC

life.augmented

The RNG peripheral is based on continuous analog noise that provides a random 32-bit value. It is NIST SP800-90B certifiable, with a guaranteed entropy of 128 bits. In this configuration RNG delivers 16 bytes of true random bits as indicated.
The Data Ready flag is set in the status register when a set of new random data is ready and validated. It must always be used.
The RNG automatically performs NIST SP800-90B compliant health tests on the noise source (a Seed Error Current status flag is set in the case of an error).
A Clock Error Current status flag is set if the RNG clock is less than HCLK clock divided by 32. This check can be disabled, especially when the RNG clock is initialized low for maximum entropy.
An interrupt source can also be enabled to indicate an abnormal seed sequence or frequency error.
The RNG is used to feed random seeds to PKA and SAES side-channel resistant peripherals each time PKA and SAES are released from reset.



# RNG block diagram

AHB_CLK

RNG

PKA

Private bus, for side-channel protections

128-bit

Digital conditioning

4x 32-bit FIFO

32-bit random data register

Clock checker

Post process

1-bit

Health tests

RNG_CLK

Divider

Sampling

Analog seed

True RNG

Flags DRDY SECS* SEIS CECS CEIS

Interrupt Enable

IE

RNG interrupt to NVIC

| | |
|---|---|
| SEIS: | Seed error interrupt status |
| CEIS: | Clock error interrupt status |
| SECS: | Seed error current status |
| CECS: | Clock error current status |
| DRDY: | Data ready |

*(*) read-only current status*

life.augmented

5

This RNG block diagram explains how the peripheral generates random numbers, in accordance with the NIST SP800-90B specification.

RNG has two clock domains: one for the sampling of the analog source of entropy, and one for the conditioning of those raw samples and retrieval via the AHB bus.

An additional private bus has been added to initialize side-channel protections in the RNG and PKA peripherals. The RNG kernel clock has a dedicated divider inside the module.

The Data Ready flag (DRDY) is triggered as soon as the data FIFO is full and is automatically reset when no more data can be read back from the RNG.

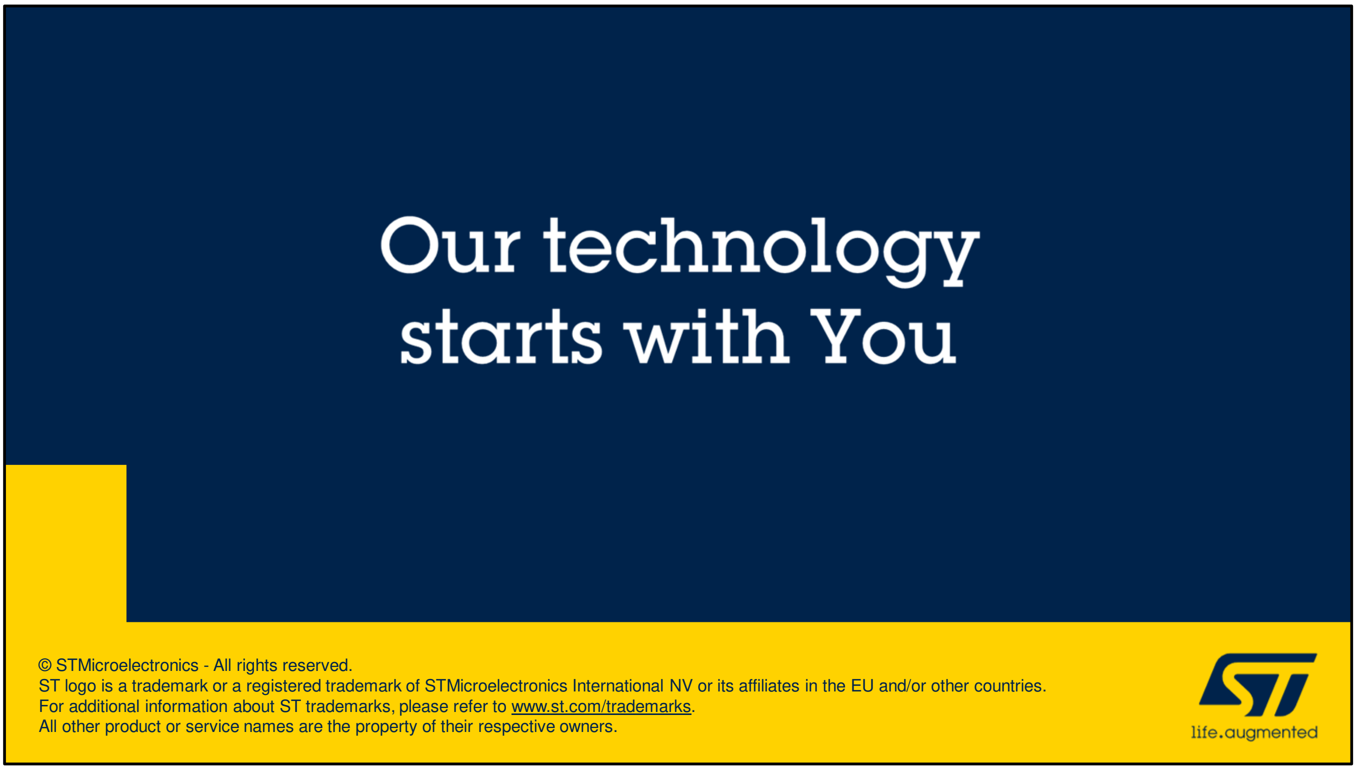
Clock checker and NIST compliant health test logic run in parallel, triggering dedicated error signals if an abnormal sequence is detected in the seed or if the RNG frequency is too low.

The TRNG block must be properly initialized with the following sequence:

1) Set the conditioning soft reset bit, CONDRST, and the correct RNG configuration in the RNG_CR register.
2) Perform a second write to the RNG_CR register with the CONDRST bit set to 0, the interrupt enable bit, IE, set to 1 and the RNG enable

bit, RNGEN,  set to1.

An interrupt is now generated when a random number is ready or when an error occurs.



# Our technology starts with You

Thank you for attending this presentation.
For more details and additional information, refer to the User Manual STM32 Cryptographic Library