53 HAL RNG Generic Driver

53.1 RNG Firmware driver registers structures

53.1.1 RNG_InitTypeDef

Data Fields

- ***uint32_t ClockErrorDetection***

Field Documentation

- ***uint32_t RNG_InitTypeDef::ClockErrorDetection***
  Clock error detection

53.1.2 NG_HandleTypeDef

Data Fields

- ***RNG_TypeDef * Instance***
- ***RNG_InitTypeDef Init***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_RNG_StateTypeDef State***
- ***uint32_t RandomNumber***

Field Documentation

- ***RNG_TypeDef* RNG_HandleTypeDef::Instance***
  Register base address
- ***RNG_InitTypeDef RNG_HandleTypeDef::Init***
  RNG configuration parameters
- ***HAL_LockTypeDef RNG_HandleTypeDef::Lock***
  RNG locking object
- ***__IO HAL_RNG_StateTypeDef RNG_HandleTypeDef::State***
  RNG communication state
- ***uint32_t RNG_HandleTypeDef::RandomNumber***
  Last Generated RNG Data

53.2 RNG Firmware driver API description

53.2.1 How to use this driver

The RNG HAL driver can be used as follows:

1. Enable the RNG controller clock using __HAL_RCC_RNG_CLK_ENABLE() macro in HAL_RNG_MspInit().
2. Activate the RNG peripheral using HAL_RNG_Init() function.
3. Wait until the 32-bit Random Number Generator contains a valid random data using (polling/interrupt) mode.
4. Get the 32 bit random number using HAL_RNG_GenerateRandomNumber() function.

53.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize the RNG according to the specified parameters in the RNG_InitTypeDef and create the associated handle

- DeInitialize the RNG peripheral
- Initialize the RNG MSP (MCU Specific Package)
- DeInitialize the RNG MSP

This section contains the following APIs:

- ***HAL_RNG_Init()***
- ***HAL_RNG_DeInit()***
- ***HAL_RNG_MspInit()***
- ***HAL_RNG_MspDeInit()***

53.2.3 Peripheral Control functions

This section provides functions allowing to:

- Get the 32 bit Random number
- Get the 32 bit Random number with interrupt enabled
- Handle RNG interrupt request

This section contains the following APIs:

- ***HAL_RNG_GenerateRandomNumber()***
- ***HAL_RNG_GenerateRandomNumber_IT()***
- ***HAL_RNG_IRQHandler()***
- ***HAL_RNG_GetRandomNumber()***
- ***HAL_RNG_GetRandomNumber_IT()***
- ***HAL_RNG_ReadLastRandomNumber()***
- ***HAL_RNG_ReadyDataCallback()***
- ***HAL_RNG_ErrorCallback()***

53.2.4 Peripheral State functions

This subsection permits to get in run-time the status of the peripheral.

This section contains the following APIs:

- ***HAL_RNG_GetState()***

53.2.5 Detailed description of functions

HAL_RNG_Init

Function name	**HAL_StatusTypeDef HAL_RNG_Init (RNG_HandleTypeDef * hrng)**
Function description	Initialize the RNG peripheral and initialize the associated handle.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **HAL:** status

HAL_RNG_DeInit

Function name	**HAL_StatusTypeDef HAL_RNG_DeInit (RNG_HandleTypeDef * hrng)**
Function description	DeInitialize the RNG peripheral.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **HAL:** status

HAL_RNG_MspInit

Function name	**void HAL_RNG_MspInit (RNG_HandleTypeDef * hrng)**
Function description	Initialize the RNG MSP.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **None:**

HAL_RNG_MspDeInit

Function name	**void HAL_RNG_MspDeInit (RNG_HandleTypeDef * hrng)**
Function description	DeInitialize the RNG MSP.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **None:**

HAL_RNG_GetRandomNumber

Function name	**uint32_t HAL_RNG_GetRandomNumber (RNG_HandleTypeDef * hrng)**
Function description	Return generated random number in polling mode (Obsolete).
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure that contains the configuration information for RNG.
Return values	• **random:** value
Notes	• Use HAL_RNG_GenerateRandomNumber() API instead.

HAL_RNG_GetRandomNumber_IT

Function name	**uint32_t HAL_RNG_GetRandomNumber_IT (RNG_HandleTypeDef * hrng)**
Function description	Return a 32-bit random number with interrupt enabled (Obsolete).
Parameters	• **hrng:** RNG handle
Return values	• **32-bit:** random number
Notes	• Use HAL_RNG_GenerateRandomNumber_IT() API instead.

HAL_RNG_GenerateRandomNumber

Function name	**HAL_StatusTypeDef HAL_RNG_GenerateRandomNumber (RNG_HandleTypeDef * hrng, uint32_t * random32bit)**
Function description	Generate a 32-bit random number.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure. • **random32bit:** pointer to generated random number variable if successful.
Return values	• **HAL:** status
Notes	• Each time the random number data is read the RNG_FLAG_DRDY flag is automatically cleared.

HAL_RNG_GenerateRandomNumber_IT

Function name	**HAL_StatusTypeDef HAL_RNG_GenerateRandomNumber_IT (RNG_HandleTypeDef * hrng)**
Function description	Generate a 32-bit random number in interrupt mode.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **HAL:** status

HAL_RNG_ReadLastRandomNumber

Function name	**uint32_t HAL_RNG_ReadLastRandomNumber (RNG_HandleTypeDef * hrng)**
Function description	Read latest generated random number.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **random:** value

HAL_RNG_IRQHandler

Function name	**void HAL_RNG_IRQHandler (RNG_HandleTypeDef * hrng)**
Function description	Handle RNG interrupt request.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **None:**
Notes	• In the case of a clock error, the RNG is no more able to generate random numbers because the PLL48CLK clock is not correct. User has to check that the clock controller is correctly configured to provide the RNG clock and clear the CEIS bit using __HAL_RNG_CLEAR_IT(). The clock error has no impact on the previously generated random numbers, and the RNG_DR register contents can be used. • In the case of a seed error, the generation of random numbers is interrupted as long as the SECS bit is '1'. If a number is available in the RNG_DR register, it must not be used because it may not have enough entropy. In this case, it is recommended to clear the SEIS bit using __HAL_RNG_CLEAR_IT(), then disable and enable the RNG peripheral to reinitialize and restart the RNG. • User-written HAL_RNG_ErrorCallback() API is called once whether SEIS or CEIS are set.

HAL_RNG_ErrorCallback

Function name	**void HAL_RNG_ErrorCallback (RNG_HandleTypeDef * hrng)**
Function description	RNG error callback.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **None:**

HAL_RNG_ReadyDataCallback

Function name	**void HAL_RNG_ReadyDataCallback (RNG_HandleTypeDef * hrng, uint32_t random32bit)**
Function description	Data Ready callback in non-blocking mode.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure. • **random32bit:** generated random value
Return values	• **None:**

HAL_RNG_GetState

Function name	**HAL_RNG_StateTypeDef HAL_RNG_GetState (RNG_HandleTypeDef * hrng)**
Function description	Return the RNG handle state.
Parameters	• **hrng:** pointer to a RNG_HandleTypeDef structure.
Return values	• **HAL:** state

53.3 RNG Firmware driver defines

53.3.1 RNG

RNG Clock Error Detection

RNG_CED_ENABLE	Clock error detection enabled
RNG_CED_DISABLE	Clock error detection disabled

RNG Exported Macros

__HAL_RNG_RESET_HANDLE_STATE	**Description:** • Reset RNG handle state. **Parameters:** • __HANDLE__: RNG Handle **Return value:** • None
__HAL_RNG_ENABLE	**Description:** • Enable the RNG peripheral. **Parameters:** • __HANDLE__: RNG Handle **Return value:** • None
__HAL_RNG_DISABLE	**Description:** • Disable the RNG peripheral. **Parameters:** • __HANDLE__: RNG Handle