Arm® True Random Number Generator (TRNG)

Revision: r0p0

Technical Reference Manual

arm

100976_0000_01_en

Arm® True Random Number Generator (TRNG)

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	05 May 2017	Non-Confidential	First release for r0p0
0000-01	05 May 2020	Non-Confidential	Second release for r0p0

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

www.arm.com

Contents
Arm® True Random Number Generator (TRNG) Technical Reference Manual

Preface

This preface introduces the *Arm® True Random Number Generator (TRNG) Technical Reference Manual*.

It contains the following:

About this book

This book is for the Arm® True Random Number Generator (TRNG).

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the TRNG.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

This chapter provides an introduction to the Arm True Random Number Generator (TRNG).

Chapter 2 Functional description

This chapter describes the functions of the Arm True Random Number Generator (TRNG).

Chapter 3 Programmers model

This chapter describes the TRNG register addresses and functionality for integration.

Chapter 4 Signal descriptions

This chapter describes the top-level signals of the True Random Number Generator (TRNG).

Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic

Introduces special terminology, denotes cross-references, and citations.

bold

Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`mono`space

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`

Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`
Denotes language keywords when used outside example code.

`<and>`
Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

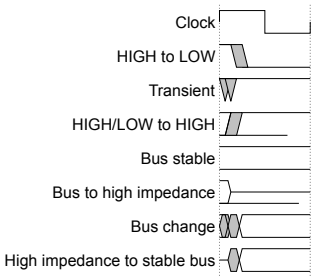

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level
The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name, n denotes an active-LOW signal.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm Publications

Document name	Document ID	Licensee only Y/N
Arm® AMBA® APB Protocol Specification	IHI 0024C	N
Arm® TRNG Characterization Application Note	100685	N
Arm® True Random Number Generator (TRNG) Configuration and Integration Manual	100977	Y
Arm® True Random Number Generator (TRNG) Software Integrators Manual	101049	N

Note

The *Arm® TRNG Characterization Application Note* and the *Arm® True Random Number Generator (TRNG) Software Integrators Manual* are available on GitHub (*https://github.com/ARM-software/TZ-TRNG/*).

Other Publications

Document ID	Document name
BSI AIS-31	*Functionality Classes and Evaluation Methodology for True Random Number Generators*
NIST SP 800-90B	*Recommendation for the Entropy Sources Used for Random Bit Generation*
FIPS Publication 140-2	*Security Requirements for Cryptographic Modules*

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Arm True Random Number Generator (TRNG) Technical Reference Manual*.
- The number 100976_0000_01_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

——— **Note** ———

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

———————————

Chapter 1
Introduction

This chapter provides an introduction to the Arm True Random Number Generator (TRNG).

It contains the following sections:

1.1 About the TRNG

The True Random Number Generator (TRNG) generates a random bit stream.

The entropy of the random bit stream complies with the standards that are described in *1.2 Compliance on page 1-13*. The TRNG is designed for simple SoC integration. The typical usage of a TRNG is key generation or for seeding approved deterministic random number generators.

1.2 Compliance

The True Random Number Generator (TRNG) complies with, or implements, the following specifications:

Table 1-1 TRNG compliance

Document ID	Document name	Compliance
FIPS Publication 140-2	*Security Requirements for Cryptographic Modules*	The TRNG complies with all applicable true random number generator requirements.
BSI AIS-31	*Functionality Classes and Evaluation Methodology for True Random Number Generators*	Fully compliant.
NIST SP 800-90B	*Recommendation for the Entropy Sources Used for Random Bit Generation*	Fully compliant.
IHI 0024C	*Arm® AMBA® APB Protocol Specification*	The TRNG complies with the APB2 protocol.

1.3 Features

The TRNG generates a random bit stream.

The TRNG core has the following key features:

- Produces 10K bits/second of entropy when core is running at 200MHz.
- Includes an internal entropy source that is based on a chain of digital inverters. The inverter cells are taken from a standard cell library. No special cells are required.
  - Odd number of inverters, leading to continuous oscillation (while active).
- Built-in hardware tests for auto correlation and *Continuous Random Number Generation Testing* (CRNGT) as required by the following standards:
  - FIPS 140-2, *Security Requirements for Cryptographic Modules*.
  - AIS-31, *Functionality Classes and Evaluation Methodology for True Random Number Generators*.
- AMBA APB2 slave interface.

1.4 Product design flow and documentation

Arm recommends that you perform some of the implementation stages before integrating TRNG into your wider SoC.

Implementation
The implementer configures the IP, replaces generic cells, and synthesizes the RTL.

Integration
The integrator connects the implemented design into an SoC. Integration includes connecting the design to a memory system, processors, and peripherals.

Final SoC implementation
This stage is the process of implementing the final, fully integrated SoC in silicon. Arm can provide only guidance relevant to its own products for this process. If Arm provides guidance on this process for your product, then a separate document is included in the implementation bundle for that product.

This section contains the following subsection:
- *1.4.1 Documentation* on page 1-15.

1.4.1 Documentation

Each TRNG document has an intended audience and is associated with specific tasks in the design flow.

For relevant protocol and architectural information that relates to this product, see *Additional reading* on page 8.

The TRNG documentation is as follows:

Technical Reference Manual
The Technical Reference Manual (TRM) describes the functionality and the effects of functional options on the behavior of TRNG. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that are described in the TRM are not relevant. If you are programming TRNG, then contact the implementer to determine:
- The build configuration of the implementation.
- What integration, if any, was performed before implementing the TRNG.

Configuration and Integration Manual
The Configuration and Integration Manual (CIM) describes:
- A list of the design-time configuration options.
- The available build configuration options and related issues in selecting them.
- How to configure the Register Transfer Level (RTL) with the build configuration options.
- How to integrate RAM arrays.
- How to run test vectors.
- The processes to sign off the configured design.

The Arm product deliverables include reference scripts and information about using them to implement your design. Reference methodology flows supplied by Arm are example reference implementations.

The CIM is a confidential book that is only available to licensees.

TRNG Characterization Application Note

The TRNG Characterization Application Note (TCAN) describes the characterization procedure that you must follow.

Software Integrators Manual
The Software Integrators Manual (SIM) describes:
- Software features and capabilities.
- Software integration guidelines.

The SIM is a confidential book that is only available to licensees.

1.5 Product revisions

The differences in functionality between the TRNG product revisions are:

r0p0 First release.

Chapter 2
Functional description

This chapter describes the functions of the Arm True Random Number Generator (TRNG).

It contains the following section:

2.1 Interfaces

The TRNG has several interfaces.

The following figure illustrates a top view of the TRNG and its interfaces.

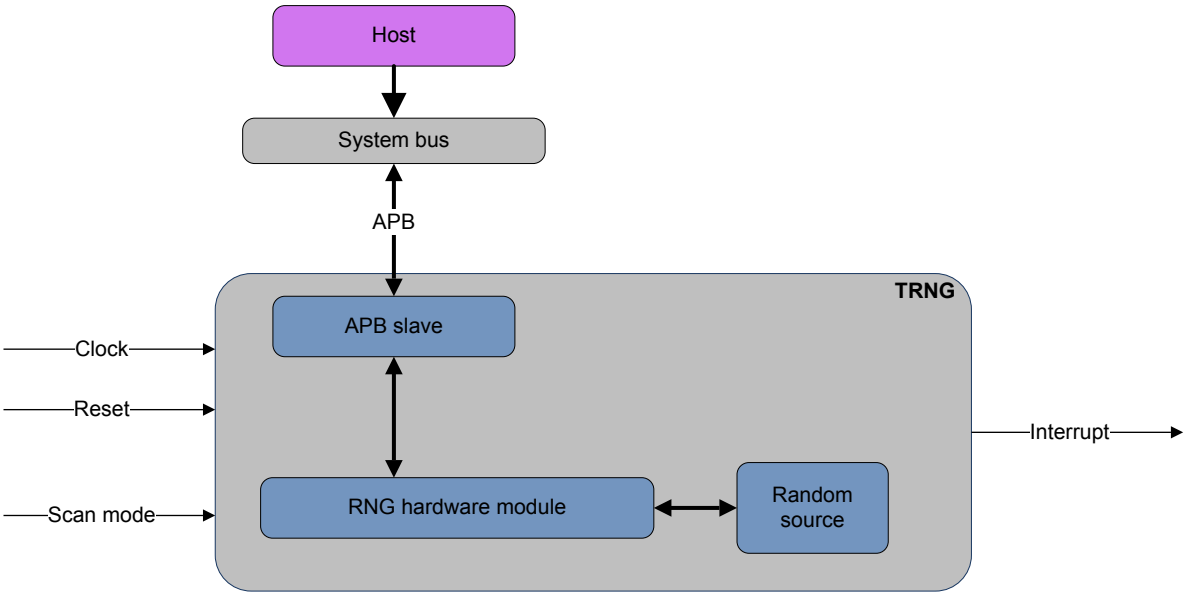

Figure 2-1 TRNG hardware overview

This section contains the following subsections:

- *2.1.1 APB slave interface* on page 2-19.
- *2.1.2 Clock interface* on page 2-19.
- *2.1.3 Reset interface* on page 2-19.
- *2.1.4 Interrupt interface* on page 2-19.
- *2.1.5 Scan interface* on page 2-20.

2.1.1 APB slave interface

The TRNG connects as an APB slave on the SoC system bus.

This interface enables a host processor to access the TRNG. The APB slave interface adheres to the *Arm® AMBA® APB Protocol Specification*.

Related information

ARM® AMBA® Specification (Rev 2)

2.1.2 Clock interface

The only clock for the TRNG is **rng_clk**.

There is no clock gating mechanism implemented inside the TRNG.

2.1.3 Reset interface

The reset interface consists of the **sys_rst_n** signal, the asynchronous global reset signal of the TRNG.

The reset is generated by the SoC.

2.1.4 Interrupt interface

cc_host_int_req (active-HIGH) serves as the interrupt output of the TRNG to the SoC.

The **cc_host_int_req** output remains HIGH until the SoC clears the interrupt source bits in the RNG_ICR register.

Note

You must connect this interrupt to the interrupt controller of your SoC.

2.1.5 Scan interface

The TRNG supports scan cell insertion methodology for the SoC *Design for Test* (DFT) strategy.

DFT control signals provide high coverage for test strategy for TRNG design.

Note

Avoid automatic insertion of DFT control or observation points into the `dx_inv_chain` module. This module already contains DFT control and observation points. For more information, see the *Arm® True Random Number Generator (TRNG) Configuration and Integration Manual*.

Chapter 3
Programmers model

This chapter describes the TRNG register addresses and functionality for integration.

Arm provides software with the TRNG product, to simplify the use of these registers.

Note

The TRNG product bundle includes header files for TRNG register offsets (`rng_hw_defs.h`), which are located in `arm_trng/tests/dsm_includes`.

It contains the following sections:

3.1 About the programmers model

The TRNG registers have a set size, but no fixed base address. There are four different types of access permissions.

The following information applies to the TRNG registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- All TRNG registers are 32-bit.

The following table lists the access types that *3.2 Register summary* on page 3-24 shows.

Table 3-1 Access permissions

Access type	Description
RO	Read-only.
RW	Read and write.
RWs	Read and write, but the register changes according to internal state.
WO	Write-only.

3.2 Register summary

The following table lists the registers in the TRNG.

Table 3-2 TRNG register summary

Offset	Name	Access[a]	Reset value	Description
0x000-0x0FC	-	-	-	Reserved.
0x100	RNG_IMR	RWs	0x0000000F	*3.3 RNG_IMR* on page 3-25
0x104	RNG_ISR	RO	0x00000000	*3.4 RNG_ISR* on page 3-26
0x108	RNG_ICR	WO	0x00000000	*3.5 RNG_ICR* on page 3-27
0x10C	TRNG_CONFIG	RW	0x00000000	*3.6 TRNG_CONFIG* on page 3-28
0x110	TRNG_VALID	RO	0x00000000	*3.7 TRNG_VALID* on page 3-29
0x114	EHR_DATA0	RO	0x00000000	*3.8 EHR_DATA_0* on page 3-30
0b118	EHR_DATA1	RO	0x00000000	*3.9 EHR_DATA_1* on page 3-31
0x11C	EHR_DATA2	RO	0x00000000	*3.10 EHR_DATA_2* on page 3-32
0x120	EHR_DATA3	RO	0x00000000	*3.11 EHR_DATA_3* on page 3-33
0x124	EHR_DATA4	RO	0x00000000	*3.12 EHR_DATA_4* on page 3-34
0x128	EHR_DATA5	RO	0x00000000	*3.13 EHR_DATA_5* on page 3-35
0x12C	RND_SOURCE_ENABLE	RW	0x00000000	*3.14 RND_SOURCE_ENABLE* on page 3-36
0x130	SAMPLE_CNT1	RW	0x0000FFFF	*3.15 SAMPLE_CNT1* on page 3-37
0x134	AUTOCORR_STATISTIC	RWs	0x00000000	*3.16 AUTOCORR_STATISTIC* on page 3-38
0x138	TRNG_DEBUG_CONTROL	RW	0x00000000	*3.17 TRNG_DEBUG_CONTROL* on page 3-39
0x13C	-	-	-	Reserved.
0x140	TRNG_SW_RESET	WO	0x00000000	*3.19 RNG_SW_RESET* on page 3-41
0x144-0x1B0	-	-	-	Reserved.
0x1B4	RNG_DEBUG_EN_INPUT	RW	0x00000000	*3.18 RNG_DEBUG_EN_INPUT* on page 3-40
0x1B8	RNG_BUSY	RO	0x00000000	*3.20 RNG_BUSY* on page 3-42
0x1BC	RST_BITS_COUNTER	WO	0x00000000	*3.21 RST_BITS_COUNTER* on page 3-43
0x1C0	RNG_VERSION	RO	0x00001111	*3.22 RNG_VERSION* on page 3-44
0x1C4-0x1DC	-	-	-	Reserved.
0x1E0	RNG_BIST_CNTR0	RO	0x00000000	*3.23 RNG_BIST_CNTR_0* on page 3-45
0x1E4	RNG_BIST_CNTR1	RO	0x00000000	*3.24 RNG_BIST_CNTR_1* on page 3-46
0x1E8	RNG_BIST_CNTR2	RO	0x00000000	*3.25 RNG_BIST_CNTR_2* on page 3-47
0x1EC-0x1FC	-	-	-	Reserved.

[a] See *Table 3-1 Access permissions* on page 3-23 for more information.

3.3 RNG_IMR

The interrupt masking register (RNG_IMR) enables you to prevent the assertion of the interrupt output.

Note

See *3.4 RNG_ISR* on page 3-26 for explanation on the interrupts that are being masked by this register.

The RNG_IMR register bit assignments are:

[0] EHR_VALID_INT_MASK

1'b1 - masks the EHR interrupt. No interrupt is generated.

[1] AUTOCORR_ERR_INT_MASK

1'b1 - masks the autocorrelation interrupt. No interrupt is generated.

[2] CRNGT_ERR_INT_MASK

1'b1 - masks the CRNGT error interrupt. No interrupt is generated.

[3] VN_ERR_INT_MASK

1'b1 - masks the von Neumann error interrupt. No interrupt is generated.

[31:4] RESERVED

Reserved

3.4 RNG_ISR

The RNG_ISR register returns the status of the interrupts.

Note

If the corresponding RNG_IMR bit is unmasked, then an interrupt is generated.

The RNG_ISR register bit assignments are:

[0] EHR_VALID

Set to 1 when 192 bits have been collected in the TRNG, and the EHR_DATA[0, 1, 2,…5] registers are ready to be read.

[1] AUTOCORR_ERR

When set to 1, it indicates that the Autocorrelation test failed four times in a row.

When set, the TRNG stops functioning until the next reset.

[2] CRNGT_ERR

When set to 1, it indicates a *Continuous Random Number Generation Testing* (CRNGT) error in the TRNG test failed.

Failure occurs when two consecutive blocks of 16 collected bits are equal.

[3] VN_ERR

When set to 1 it indicates a von Neumann error.

A von Neumann error occurs if 32 consecutive collected bits are identical, that is, 32 zeros or 32 ones.

[31:4] RESERVED

Reserved.

3.5 RNG_ICR

The RNG_ICR register enables the host processor to clear the interrupts.

The RNG_ICR register bit assignments are:

[0] EHR_VALID

Set to 1 after the EHR_DATA[0,1,2,…5] registers have been read.

[1] AUTOCORR_ERR

Software cannot clear this bit. Only a TRNG reset can clear this bit.

[2] CRNGT_ERR

Set to 1, to clear a *Continuous Random Number Generation Testing* (CRNGT) error.

[3] VN_ERR

Set to 1, to clear a von Neumann error.

[31:4] RESERVED

Reserved

3.6 TRNG_CONFIG

This register handles the TRNG configuration.

The TRNG_CONFIG register bit assignments are:

[1:0] RND_SRC_SEL

Defines the length of the oscillator ring (= the number of inverters) out of four possible selections.

[31:2] RESERVED

Reserved

3.7 TRNG_VALID

This register indicates that the TRNG data is valid.

The TRNG_VALID register bit assignments are:

[0] EHR_VALID

1'b1 indicates that collection of bits in the TRNG is completed, and data can be read from the EHR_DATA registers.

[31:1] RESERVED

Reserved

3.8 EHR_DATA_0

This register returns 32 bits from the 192-bit Entropy Holding Register, containing the generated random number.

——— **Note** ———

The EHR_DATA registers can only be set while the TRNG is in debug mode (**rng_debug_enable** input is set).

———————————

The EHR_DATA_0 register bit assignments are:

[31:0] EHR_DATA

Returns bits [31:0] of the EHR.

3.9 EHR_DATA_1

This register returns 32 bits from the 192-bit Entropy Holding Register, containing the generated random number.

——— **Note** ———

The EHR_DATA registers can only be set while the TRNG is in debug mode (**rng_debug_enable** input is set).

———————————

The EHR_DATA_1 register bit assignments are:

[31:0] EHR_DATA

Returns bits [63:32] of the EHR.

3.10 EHR_DATA_2

This register returns 32 bits from the 192-bit Entropy Holding Register, containing the generated random number.

——— **Note** ———

The EHR_DATA registers can only be set while the TRNG is in debug mode (**rng_debug_enable** input is set).

———————————

The EHR_DATA_2 register bit assignments are:

[31:0] EHR_DATA

Returns bits [95:64] of the EHR.

3.11 EHR_DATA_3

This register returns 32 bits from the 192-bit Entropy Holding Register, containing the generated random number.

——— **Note** ———

The EHR_DATA registers can only be set while the TRNG is in debug mode (**rng_debug_enable** input is set).

———————————

The EHR_DATA_3 register bit assignments are:

[31:0] EHR_DATA

Returns bits [127:96] of the EHR.

3.12 EHR_DATA_4

This register returns 32 bits from the 192-bit Entropy Holding Register, containing the generated random number.

——— **Note** ———

The EHR_DATA registers can only be set while the TRNG is in debug mode (**rng_debug_enable** input is set).

———————————

The EHR_DATA_4 register characteristics are:

[31:0] EHR_DATA

Returns bits [159:128] of the EHR.

3.13 EHR_DATA_5

This register returns 32 bits from the 192-bit Entropy Holding Register, containing the generated random number.

——— **Note** ———

The EHR_DATA registers can only be set while the TRNG is in debug mode (**rng_debug_enable** input is set).

———————————

The EHR_DATA_5 register characteristics are:

[31:0] EHR_DATA

Returns bits [191:160] of the EHR.

3.14 RND_SOURCE_ENABLE

This register holds the enable signal for the random source.

The RND_SOURCE_ENABLE register bit assignments are:

[0] RND_SRC_EN

The enable signal for the random source.

[31:1] RESERVED

Reserved

3.15 SAMPLE_CNT1

The SAMPLE_CNT1 register stores the number of rng_clk cycles between two consecutive ROSC samples.

The SAMPLE_CNT1 register characteristics are:

[31:0] SAMPLE_CNTR1

Sets the number of rng_clk cycles between two consecutive ring oscillator samples.

——— **Note** ———

If the von Neumann balancer is bypassed, the minimum value for the sample counter must not be less than decimal 17.

———————————

3.16 AUTOCORR_STATISTIC

The AUTOCORR_STATISTIC register returns statistics about autocorrelation test activations.

The AUTOCORR_STATISTIC register bit assignments are:

[13:0] AUTOCORR_TRYS

Counts each time an autocorrelation test starts. Any write to the register resets the counter. Stops collecting statistics if one of the counters has reached the limit.

[21:14] AUTOCORR_FAILS

Counts each time an autocorrelation test fails. Any write to the register resets the counter. Stops collecting statistics if one of the counters has reached the limit.

[31:22] RESERVED

Reserved

3.17 TRNG_DEBUG_CONTROL

The TRNG_DEBUG_CONTROL register controls the debug behavior of the TRNG.

——— **Note** ———

These bits can only be set while in debug mode, unless the TRNG_TESTS_BYPASS_EN HW flag is defined.

———————————

The TRNG_DEBUG_CONTROL register bit assignments are:

[0] RESERVED

Reserved

[1] VNC_BYPASS

When this bit is set, the von Neumann balancer is bypassed (including the 32 consecutive bits test).

[2] TRNG_CRNGT_BYPASS

When this bit is set, the CRNGT test in the TRNG is bypassed.

[3] AUTO_CORRELATE_BYPASS

When this bit is set, the autocorrelation test in the TRNG module is bypassed.

[31:4] RESERVED

Reserved

3.18 RNG_DEBUG_EN_INPUT

The RNG_DEBUG_INPUT defines the RNG in debug mode.

The RNG_DEBUG_EN_INPUT register bit assignments are:

[0] RNG_DEBUG_EN

Reflects the rng_debug_enable input port

[31:1] RESERVED

Reserved

——— **Note** ———

The rng_debug_enable input port is tied to 0 in `rng_top_wrap_unconnected.v`.

———————————

3.19 RNG_SW_RESET

The TRNG_SW_RESET register enables SW to reset the TRNG.

The RNG_SW_RESET register bit assignments are:

[0] RNG_SW_RESET

Any value written (1'b0 or 1'b1) causes a reset cycle to the TRNG block. The reset mechanism takes about four RNG clock cycles until the reset line is deasserted.

[31:1] RESERVED

Reserved

3.20 RNG_BUSY

The RNG_BUSY register indicates when the TRNG is busy.

The RNG_BUSY register bit assignments are:

[0] RNG_BUSY

Reflects the status of the **rng_busy** signal.

[31:1] RESERVED

Reserved.

3.21 RST_BITS_COUNTER

The RST_BITS_COUNTER resets the counter of collected bits in the TRNG.

The RST_BITS_COUNTER register bit assignments are:

[0] RST_BITS_COUNTER

Writing any value to this address resets the bits counter and TRNG_VALID registers. The RND_SOURCE_ENABLE register must be unset in order for reset to take place.

[31:1] RESERVED

Reserved

——— **Note** ———

Since the **rng_debug_enable** input port is tied to 0 in `rng_top_wrap_unconnected.v`, writing the RST_BIT_COUNTER register has no effect.

———————————

3.22 RNG_VERSION

The RNG_VERSION register displays the version settings of the TRNG. This register is read-only.

The RNG_VERSION register bit assignments are:

[0] EHR_WIDTH_192

- 1'b0 - 128-bit EHR
- 1'b1 - 192-bit EHR

[1] CRNGT_EXISTS

- 1'b0 - Does not exist.
- 1'b1 - Exists.

[2] AUTOCORR_EXISTS

- 1'b0 - Does not exist.
- 1'b1 - Exists

[3] TRNG_TESTS_BYPASS_EN

- 1'b0 - TRNG tests bypass not enabled.
- 1'b1 - TRNG tests bypass enabled.

[4] PRNG_EXISTS

- 1'b0 - Does not exist.
- 1'b1 - Exists.

[5] KAT_EXISTS

- 1'b0 - Does not exist.
- 1'b1 - Exists.

[6] RESEEDING_EXISTS

- 1'b0 - Does not exist.
- 1'b1 -Exists.

[7] RNG_USE_5_SBOXES

- 1'b0 - 20 SBOX AES
- 1'b1 - 5 SBOX AES

[31:8] RESERVED

Reserved

The reset value of RNG_VERSION is `0x00001111`.

Note

The PRNG_EXISTS, KAT_EXISTS, RESEEDING_EXISTS, RNG_USE_5_SBOXES fields are related to legacy features that are no longer supported by Arm. They should not be enabled by defining the corresponding Verilog macros in the RTL.

3.23 RNG_BIST_CNTR_0

The RNG_BIST_CNTR_0 register returns the collected BIST results.

The RNG_BIST_CNTR_0 bit assignments are:

[21:0] ROSC_CNTR_VAL

Returns the results of the TRNG BIST counter.

[31:22] RESERVED

Reserved.

3.24 RNG_BIST_CNTR_1

The RNG_BIST_CNTR_1 registers returns the collected BIST results.

The RNG_BIST_CNTR_1 bit assignments are:

[21:0] ROSC_CNTR_VAL

Returns the results of the TRNG BIST counter.

[31:22] RESERVED

Reserved.

3.25 RNG_BIST_CNTR_2

The RNG_BIST_CNTR_2 registers returns the collected BIST results.

The RNG_BIST_CNTR_2 bit assignments are:

[21:0] ROSC_CNTR_VAL

Returns the results of the TRNG BIST counter.

[31:22] RESERVED

Reserved.

Chapter 4
Signal descriptions

This chapter describes the top-level signals of the True Random Number Generator (TRNG).

It contains the following sections:

4.1 Clocks and resets

The clock and reset input signals include the asynchronous global reset and the TRNG engine clock.

Table 4-1 Clock and reset signals

Signal	Direction	Description	Connection information
sys_rst_n	Input	Asynchronous global reset (active-LOW).	Connect to a reset controller.
rng_clk	Input	TRNG engine clock.	Connect to a clock generator.

4.2 APB slave interface

The APB slave interface enables a host processor to access the TRNG.

The following table lists the AMBA APB2 slave interface signals.

Table 4-2 APB slave interface signals

Signal	Direction	Description	Connection information
cc_psel	Input	Peripheral select signal.	Connect to an APB master device.
cc_penable	Input	Indicates that the enable cycle is taking place.	
cc_paddr[11:0]	Input	Peripheral address bus. The TRNG address space requires 4KB.	
cc_pwrite	Input	Peripheral write signal.	
cc_pwdata[31:0]	Input	Peripheral write data bus.	
cc_prdata[31:0]	Output	Peripheral read data bus.	

4.3 Interrupts

The following table lists the interrupt signals.

Table 4-3 Interrupt signals

Signal	Direction	Description	Connection information
cc_host_int_req	Output	An active-HIGH interrupt that remains HIGH until the host processor writes to the Interrupt Clear Register, RNG_ICR.	Connect to an interrupt controller or the host processor.

Related references

3.4 RNG_ISR on page 3-26
3.3 RNG_IMR on page 3-25
3.5 RNG_ICR on page 3-27

4.4 Scan

The following table lists the scan test signals.

Table 4-4 Scan signals

Signal	Direction	Description	Connection information
scanmode	Input	**scanmode** must be: • HIGH throughout the duration of scan testing. • LOW during functional mode.	Connect to your SoC scan logic.

Appendix A
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

A.1 Revisions "TRNG" TRM

This appendix describes changes between released issues of this book.

Table A-1 Issue 0000-00

Change	Location	Affects
First release	-	-

Table A-2 Differences between issue 0000-00 and issue 0000-01

Change	Location	Affects
Removed TrustZone® from product name	Entire document	All revisions
Changed the product description in *1.1 About the TRNG* on page 1-12	*1.1 About the TRNG* on page 1-12	All revisions
The TRNG_DEBUG_CONTROL register changed from RO to RW	*3.2 Register summary* on page 3-24	All revisions
Added information from RTL to *3.15 SAMPLE_CNT1* on page 3-37	*3.15 SAMPLE_CNT1* on page 3-37	All revisions
Added the chapter *3.18 RNG_DEBUG_EN_INPUT* on page 3-40 to *Chapter 3 Programmers model* on page 3-21	*Chapter 3 Programmers model* on page 3-21	All revisions
Added RNG_DEBUG_EN_INPUT to *3.2 Register summary* on page 3-24	*3.2 Register summary* on page 3-24	All revisions
Added *3.22 RNG_VERSION* on page 3-44 to *Chapter 3 Programmers model* on page 3-21	*3.22 RNG_VERSION* on page 3-44	All revisions
Added RNG_VERSION to *3.2 Register summary* on page 3-24	*3.2 Register summary* on page 3-24	All versions
Renamed the section 'About Functions" and added a more detailed description to each interface	*Chapter 2 Functional description* on page 2-18	All revisions
Added *2.1.1 APB slave interface* on page 2-19 to *Chapter 2 Functional description* on page 2-18	*2.1.1 APB slave interface* on page 2-19	All versions
Added *2.1.2 Clock interface* on page 2-19 to *Chapter 2 Functional description* on page 2-18	*2.1.2 Clock interface* on page 2-19	All versions
2.1.3 Reset interface on page 2-19 to *Chapter 2 Functional description* on page 2-18	*2.1.3 Reset interface* on page 2-19	All versions
2.1.4 Interrupt interface on page 2-19 to *Chapter 2 Functional description* on page 2-18	*2.1.4 Interrupt interface* on page 2-19	All versions
2.1.5 Scan interface on page 2-20 to *Chapter 2 Functional description* on page 2-18	*2.1.5 Scan interface* on page 2-20	All versions
Added a note to the *Chapter 3 Programmers model* on page 3-21	*Chapter 3 Programmers model* on page 3-21	All versions
Expanded EHR_DATA into *3.8 EHR_DATA_0* on page 3-30, *3.9 EHR_DATA_1* on page 3-31,*3.10 EHR_DATA_2* on page 3-32,*3.11 EHR_DATA_3* on page 3-33,*3.12 EHR_DATA_4* on page 3-34, and *3.13 EHR_DATA_5* on page 3-35	*Chapter 3 Programmers model* on page 3-21	All versions
Expanded RNG_BIST_CNTR into *3.23 RNG_BIST_CNTR_0* on page 3-45, *3.24 RNG_BIST_CNTR_1* on page 3-46, and *3.25 RNG_BIST_CNTR_2* on page 3-47	*Chapter 3 Programmers model* on page 3-21	All versions
Changed RESERVED value to [31:1] *3.20 RNG_BUSY* on page 3-42.	*3.20 RNG_BUSY* on page 3-42	All versions

Table A-2 Differences between issue 0000-00 and issue 0000-01 (continued)

Change	Location	Affects
Changed **rst_n** to **syst_rst_n** in *4.1 Clocks and resets* on page 4-49 and *2.1.3 Reset interface* on page 2-19.	Entire document	All versions
Changed RESERVED register value from `0x144-0x1B4` to `0x144-0x1B0` in *3.2 Register summary* on page 3-24	*3.2 Register summary* on page 3-24	All versions
Added directory for product bundle in *Chapter 3 Programmers model* on page 3-21	*Chapter 3 Programmers model* on page 3-21	All versions
Added connections to the tables in *Chapter 4 Signal descriptions* on page 4-48	*Chapter 4 Signal descriptions* on page 4-48	All versions
Added directions to **cc_penable**, **cce_paddr[11:0]**, **cc_pwrite**, and **cc_pwdata[31:0]** in *2.1.1 APB slave interface* on page 2-19	*2.1.1 APB slave interface* on page 2-19	All versions
Added detail to the description of *4.4 Scan* on page 4-52	*4.4 Scan* on page 4-52	All versions