



Arm[®] True Random Number Generator (TRNG)

Revision: r0p0

Characterization Application Note

Non-Confidential

Copyright © 2015–2016, 2018–2020, 2022 Arm Limited (or its affiliates).
All rights reserved.

Issue 11

100685_0000_11_en



Arm® True Random Number Generator (TRNG) Characterization Application Note

Copyright © 2015–2016, 2018–2020, 2022 Arm Limited (or its affiliates). All rights reserved.

Release Information

Document history

| Issue | Date | Confidentiality | Change |
|---------|------------------|------------------|--------------------------------|
| 00 | 27 July 2015 | Confidential | First official release (v1.0). |
| 0000-01 | 8 October 2015 | Confidential | Second release (v1.1). |
| 0000-02 | 22 November 2015 | Confidential | Third release (v1.2). |
| 0000-03 | 27 January 2016 | Confidential | Fourth release (v1.3). |
| 0000-04 | 17 May 2016 | Confidential | Fifth release (v1.4). |
| 0000-05 | 8 November 2016 | Confidential | Sixth release. |
| 0000-06 | 9 January 2018 | Non-Confidential | Seventh release. |
| 0000-07 | 6 June 2018 | Non-Confidential | Eighth release. |
| 0000-08 | 6 September 2018 | Non-Confidential | Ninth release. |
| 0000-09 | 16 January 2019 | Non-Confidential | Tenth release. |
| 0000-10 | 5 February 2020 | Non-Confidential | Eleventh release. |
| 0000-11 | 25 March 2022 | Non-Confidential | Twelfth release. |

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2015–2016, 2018–2020, 2022 Arm Limited (or its affiliates). All rights reserved.

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>.

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

| | |
|--|-----------|
| 1 Introduction..... | 6 |
| 1.1 Conventions..... | 6 |
| 1.2 Additional reading..... | 7 |
| 1.3 Other information..... | 7 |
| 2 Overview of Arm® True Random Number Generator (TRNG)..... | 8 |
| 2.1 TRNG characterization..... | 8 |
| 2.2 Compliance..... | 9 |
| 3 TRNG characterization procedure..... | 10 |
| 3.1 Characterization procedure overview..... | 10 |
| 3.2 Characterization test program..... | 11 |
| 3.3 Characterization test conditions..... | 12 |
| 3.3.1 Output-file names..... | 13 |
| 3.4 Base iteration..... | 14 |
| 3.5 First characterization iteration..... | 15 |
| 3.6 Second characterization iteration..... | 17 |
| 3.7 Restart tests iteration..... | 20 |
| A CC_TST_TRNG output format..... | 21 |
| B Revision history..... | 22 |

1 Introduction

1.1 Conventions





The following subsections describe conventions used in Arm documents.



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: developer.arm.com/glossary.

Typographic conventions

| Convention | Use |
|--|--|
| <i>italic</i> | Citations. |
| bold | Interface elements, such as menu names. Signal names. Terms in descriptive lists, where appropriate. |
| monospace | Text that you can enter at the keyboard, such as commands, file and program names, and source code. |
| monospace bold | Language keywords when used outside example code. |
| monospace <u>underline</u> | A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name. |
| <and> | Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></pre> |
| SMALL CAPITALS | Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE . |
|  Caution | Recommendations. Not following these recommendations might lead to system failure or damage. |
|  Warning | Requirements for the system. Not following these requirements might result in system failure or damage. |
|  Danger | Requirements for the system. Not following these requirements will result in system failure or damage. |
|  Note | An important piece of information that needs your attention. |

| Convention | Use |
|--|--|
|  Tip | A useful tip that might make it easier, better or faster to perform a task. |
|  Remember | A reminder of something important that relates to the information you are reading. |

1.2 Additional reading

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

Table 1-2: Publications

| Document ID | Document name |
|------------------------|---|
| ANSI X9.31-1988 | <i>Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry (rDSA)</i> |
| BSI AIS-31 | <i>Functionality Classes and Evaluation Methodology for True Random Number Generators</i> |
| FIPS Publication 140-2 | <i>Security Requirements for Cryptographic Modules</i> |
| NIST SP 800-90A | <i>Recommendation for Random Number Generation Using Deterministic Random Bit Generators – App C.</i> |
| NIST SP 800-90B | <i>Recommendation for the Entropy Sources Used for Random Bit Generation</i> |



Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at <http://www.adobe.com>

1.3 Other information

See the Arm® website for other relevant information.

- [Arm® Developer.](#)
- [Arm® Documentation.](#)
- [Technical Support.](#)
- [Arm® Glossary.](#)

2 Overview of Arm® True Random Number Generator (TRNG)

This chapter provides an overview of the Arm® True Random Number Generator (TRNG) and its characterization.

Arm® True Random Number Generator (TRNG) collects entropy from a physical entropy source, which is a component capable of generating an unpredictable or random output bit stream. The collected entropy is used to seed the cryptographic random bits generator with a secure initial state.



Usually, the physical process used for collecting entropy is an inverter timing jitter that is collected from a dedicated on-chip free-running ring oscillator.

The TRNG can be used in one of two modes, each requiring a different driver:

FE TRNG

The operating mode of the Arm implementation of the FE TRNG driver is compliant with the *BSI AIS-31 Functionality Classes and Evaluation Methodology for True Random Number Generators* standard, as a true random number generator that outputs full-entropy bits at a relatively low rate.

800-90B TRNG

The operating mode of the Arm implementation of the 800-90B TRNG driver is compliant with the *NIST SP 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation* standard, as a true random number generator.

2.1 TRNG characterization

Arm True Random Number Generator (TRNG) configuration parameters specify the settings of the internal ring-oscillator lengths, and the output sampling rate. The parameters are device-specific.

Each silicon process has different noise and jitter characteristics. The specific SoC layout affects these characteristics. Therefore, the TRNG behavior must be characterized on the actual silicon of the device to determine the most suitable parameters. Characterizing in this way ensures that the TRNG output has maximal entropy.

Characterization must be performed during the initial post-silicon testing of the device, or whenever substantial changes are made. For example, after changes to process or respins.

2.2 Compliance

Arm® True Random Number Generator (TRNG) complies with the following specifications:

Table 2-1: Arm® True Random Number Generator (TRNG) compliance

| Document ID | Document name | Compliance |
|-----------------|---|--|
| BSI AIS-31 | <i>Functionality Classes and Evaluation Methodology for True Random Number Generators</i> | Compliant in an implementation using FETRNG driver with class PTG.2. |
| NIST SP 800-90B | <i>Recommendation for the Entropy Sources Used for Random Bit Generation</i> | Compliant with section 4.4 Approved Continuous Health Tests. |

3 TRNG characterization procedure

This chapter provides the detailed Arm characterization procedure.

3.1 Characterization procedure overview

The full characterization procedure is divided into various steps.

You then send the resulting data to Arm. Arm analyzes the results, and returns the best Arm settings to you.



Note

If you use the 800-90B mode, it is your responsibility to analyze the results of the second iteration (including the restart tests), using the tool provided on the NIST website. For more information, see [3.6 Second characterization iteration](#) on page 17.

The following table provides an overview of the steps to follow for the TRNG characterization procedure.

Table 3-1: Characterization high-level procedure steps

| Step | Owner | Execution | Comments |
|---|-------------------|--|---|
| Prepare characterization test program. | Partner | Prepare a characterization test program that uses the CC_TST_TRNG routine that Arm supplies. | See 3.2 Characterization test program on page 11 and 3.3 Characterization test conditions on page 12. |
| Base iteration: Find the minimal sample count. | Partner | Run the preliminary test to establish the minimal (base) value of the sample count. | See 3.4 Base iteration on page 13. |
| First iteration: Run first set of characterization tests. | Partner | Run a series of characterization tests under multiple test conditions. | See 3.5 First characterization iteration on page 15. Send results to Arm. |
| First iteration: Analyze first characterization test results. | Arm | Runs a set of statistical tests on the characterization output data. | Sends the partner the TRNG configuration parameters, and a set of corners to run the tests on. |
| Second iteration: Run second set of characterization tests. | Partner | <ol style="list-style-type: none"> 1. Use CC_TST_TRNG with the configuration parameters that were received in the first iteration. 2. Run the characterization tests in worst-case conditions, as provided by Arm. | See 3.6 Second characterization iteration on page 17. Partners using AIS-31 mode, must send the results to Arm. |
| Second iteration: Analyze second characterization test results. | Arm (AIS-31 mode) | Runs a set of statistical tests on the characterization output data. Use this analysis to generate the mass production Arm® True Random Number Generator (TRNG) configuration parameters. | Sends the mass production TRNG configuration parameters to the partner. Note: If the results are not good enough, repeat the second iteration. |

| Step | Owner | Execution | Comments |
|-------------------------|------------------------|--|--|
| | Partner (800-90B mode) | Analyze the results using NIST tools. | Note: If the results are not good enough, repeat the second iteration. |
| Restart tests iteration | Partner (800-90B mode) | Re-run tests using data from an entropy source that produces outputs for real-world use. | See 3.7 Restart tests iteration on page 19. |

3.2 Characterization test program

You must implement the characterization test program, using the CC_TST_TRNG API that Arm supplies.

The CC_TST_TRNG API is included in <prefix>-TRNG_Characterization.c, where <prefix> represents the product-specific part number and version. For the macros used in following sections, refer to <prefix>-TRNG_Characterization.c.



Usually, for a memory-constrained IoT scenario, the collected raw data is redirected to a peripheral, to reduce memory usage. For a device without an available peripheral, you can utilize the max delay between blocks to design a private protocol to collect the data. The max delay between blocks follows this formula:

$$max_delay = 4 * (192 * SAMPLE_CNT / RNG_CLK).$$

Example of API

```

/*
 * Collect TRNG output for characterization
 *
 * @regBaseAddress: The base address of TRNG registers in system memory map
 * @TRNGMode:      Base iteration      - TRNG_MODE_FAST
 *                First iteration     - TRNG_MODE_FAST
 *                Second iteration    - TRNG_MODE_FE compliant with BSI AIS-31
 *                                     TRNG_MODE_80090B compliant with NIST SP
800-90B
 * @roscLength:    Ring oscillator length (0 to 3)
 * @sampleCount:   Ring oscillator sample counter value
 * @buffSize:      Total buffer size covered header, sample_bits and footer.
 *                Must be between 52 and 2^24 bytes.
 *                bufsize >= header(16 bytes)+sample_bits/8 (bytes)+footer(12
bytes).
 *                For example, if 100Mbits of data need to be collected, the
minimal
 *                buffSize should be 12,500,000+28=12,500,028 bytes. To be
safe, the
 *                buffSize can be 12,501,000 bytes.
 * @callbackFunc:  Callback function to process the real output data
 *                This callbackFunc is called many times in CC_TST_TRNG.
 *                First, it is called after the 16-byte header is generated.
 *                Then, it is called repeatedly when the 24-byte EHR registers
are full.
 *                Finally, it is called after the 12-byte footer is generated.
 * @return         0 on success. Non-zero value on failure.
 */
int CC_TST_TRNG( unsigned long regBaseAddress,
                 uint32_t TRNGMode,
                 uint32_t roscLength,

```

```
uint32_t sampleCount,  
uint32_t buffSize,  
callback_TRNG callbackFunc);
```

The TRNG modes are as follows:

- TRNG_MODE_FAST - This mode is the same thing as setting TRNG_MODE to 0. Use in Base iteration, First iteration, and Restart tests iteration (for 800-90B characterization).
- TRNG_MODE_FE - This mode is the same thing as setting TRNG_MODE to 1. Use in Second iteration for BSI AI-S31 characterization.
- TRNG_MODE_80090B - This mode is the same thing as setting TRNG_MODE to 2. Use in Second iteration for NIST SP 800-90B characterization.

Definition of callback_TRNG

```
/*  
 * Callback function type define  
 * The partner must implement this function according the system resource.  
 * This function is called by CC_TST_TRNG() to process the data generated  
 * or collected by CC_TST_TRNG().  
 * @outputSize      The size of the output data  
 * @outputBuffer    The buffer of the data  
 */  
typedef void (*callback_TRNG)(uint32_t outputSize, uint8_t *outputBuffer);
```



Partners must implement the callback function according to the system resource.

3.3 Characterization test conditions

Each characterization test is executed by running the characterization test program under a combination of conditions.

These tests are described in [Characterization test conditions](#) on page 13.

It is critical that for each test:

- If using FE TRNG driver:
 - All output bits must be collected using a single contiguous execution of the test.
 - If any bits are dropped and not captured in the output file, you must rerun the test as the statistical analysis of the output is meaningless.
 - If the system does not have sufficient memory to collect all required bits in a single run, you can split each test into multiple runs. For example 100 consecutive runs, each collecting 1Mbits.

- If using 800-90B TRNG driver:
 - The output bits can be non-contiguous, and concatenation of several smaller sets of consecutive samples (generated using the same noise source) is allowed.
 - Smaller sets must contain at least 1000 samples.
 - The concatenated dataset must contain at least 1,000,000 samples.
- For both drivers, All the resulting bits are saved in the output file without any gaps.

If any bits are dropped and not captured in the output file, the test must be rerun as the statistical analysis of the output is meaningless.

Table 3-2: Characterization test conditions

| Configuration variable | Operating conditions | Filename values |
|-------------------------|---|---|
| Ring oscillator length. | The four configurable lengths that Arm® True Random Number Generator (TRNG) allows. | R0, R1, R2, R3. R0 is the shortest length and R3 is the longest. Note: In some chips, the ring oscillator might not properly work when configured to the shorter lengths. |
| Voltage. | High, typical, low. | VH, VT, VL. |
| Temperature. | High, typical, low. | TH, TT, TL. |
| CMOS process corner. | Typical, fast/fast, fast/slow, slow/fast, slow/slow. | CT, CFF, CFS, CSF, CSS. |

3.3.1 Output-file names

Output-files are named according to a standard format.

The output file for each characterization test is named according to the Filename Values column in [Characterization test conditions](#) on page 13. Output filename format is `trng_samples_R*_S*_V*_T*_C*.bin`.

For example, for the following characterization test:

- The longest ring oscillator length.
- Sample counter value of five.
- High voltage.
- Low temperature.
- Fast or slow CMOS corner.

The filename is `trng_samples_R3_S5_VH_TL_CFS.bin`.

3.4 Base iteration

The base iteration is used to find the minimum sample counter value for which Arm® True Random Number Generator (TRNG) operates properly, under typical operating conditions.

About this task

The typical operating conditions are:

- The second-longest ring-oscillator.
- Typical voltage.
- Typical temperature.
- Typical process corner.

Procedure

1. Find the minimum sample counter by calling `CC_TST_TRNG` with increasing values of `sampleCount` (starting with 1) until it exits successfully.
 - At least 200Kbits (2.0e5 bits) must be collected for the base iteration and the first characterization iteration.
 - Set the `TRNGMode` to 0 for both the NIST 80-090B and FE-TRNG driver modes.

In many systems, the test succeeds immediately (with `sampleCount=1`), which is an expected and even desirable result.

2. Use the minimum sample counter for the [3.5 First characterization iteration](#) on page 15

Example 3-1: Base iteration sample code

The following is sample code for data type, macro, and callback function definition:

```
typedef unsigned char uint8_t;
typedef unsigned int uint32_t;
/*
 * Define the buffer size for the base iteration.
 * At least 200 Kbits(25 KBytes) must be collected
 * every time EHR registers output 192 bits (24Bytes) value.
 * Therefore, set the collected data as 25008 bytes (25008/24 = 1042).
 * The full buffer size is 25008+28(header+footer)=25036 bytes.
 */
#define BASE_ITERATION_BUF_LEN 25036 //0x61CC
static uint8_t data_buf[BASE_ITERATION_BUF_LEN];
static uint32_t buffer_index = 0;

static void callback_BaseIteration(uint32_t outputSize, uint8_t *outputBuffer)
{
    if (NULL == outputBuffer)
    {
        printf("invalid input parameter!\r\n");
    }

    memcpy(data_buf+buffer_index, outputBuffer, outputSize);
    buffer_index += outputSize;
}
```

The following is sample code for finding minimum sample counter value:

```
int ret = 0;
uint32_t i = 0;
uint32_t sampleCounter = 0;

for (i=1; ;i++)
{
    ret = CC_TST_TRNG(g_CcBaseAddr,    // regBaseAddress
0,    // TRNGMode
2,    // roscLength
i,    // sampleCount
BASE_ITERATION_BUF_LEN, // buffSize
callback_BaseIteration); // callback
    buffer_index = 0;

    if (ret == 0)
        break;
}
sampleCounter = i;
```

3.5 First characterization iteration

Perform this procedure for each of the characterization test conditions combinations.

Procedure

1. Set up the test operating conditions.

2. Run CC_TST_TRNG under these conditions with each of the following *sampleCount* values:

- First set: S_{SMP1} = The minimal *sampleCount* found in [3.4 Base iteration](#) on page 13.
- Second set: S_{SMP2} = Ceiling($1.5 * (S_{SMP1}+1)-1$).
- Third set: S_{SMP3} = Ceiling($1.5 * (S_{SMP2}+1)-1$).

For example, if the minimum value of *sampleCount* is 8, then run CC_TST_TRNG under each of the 180 conditions with:

- $SMP1=8$ in the first set.
- $SMP2=13$ in the second set.
- $SMP3=20$ in the third set.



Set the *TRNGMode* to 0 for both the NIST 80-090B and FE-TRNG driver modes.

Overall, you must run a total of 540 characterization tests: (3 (sample counter values) * 4 (ring oscillator configurations) * 3 (voltages) * 3 (temperatures) * 5 (process corners)).

You must export all the data that is collected as part of the first characterization iteration from the device, using your implementation of the *callbackFunc* callback function.

The estimated total time for the characterization procedure is between 16-18 hours: 5 chips * 3 temperature conditions * (30 minutes to set each chip in each temperature condition + [30-40] minutes for running all oscillator, voltage, and sample count combinations in this temperature).

3. Save the results of each test run in the relevant output file that is named according to [3.3.1 Output-file names](#) on page 13.

When the first characterization iteration is complete, you can expect to have 540 files that are named according to the different test conditions.

4. Send the collected files that you exported from the device to Arm.

Arm analyzes the data and returns to the following parameters.

- A set of four sample counter values, one value for each ring oscillator length.
- The worst-case corners (voltage/temperature/process) that you use for the [3.6 Second characterization iteration](#) on page 17. There are several worst-case corners for each ring oscillator.

Example 3-2: Sample code for the first characterization iteration

In this example, $SMP1=1$, $SMP2=2$ and $SMP3=4$ and *callback_FirstIteration* is used as the *callback_BaseIteration* for the base iteration.

The `printData` function used in this example is a reference. Its purpose is to output data from `data_buf` buffer through an I/O channel. `CC_TST_TRNG` collects data from the device and fills up the `data_buf` buffer by calling callback function `callback_FirstIteration`.

```
void first_iteration_test()
{
    uint32_t TRNGMode = 0;
    uint32_t roscLength = 0;
    uint32_t sampleCounter = 1;
    int ret = 0;

    for (roscLength=0; roscLength<4; roscLength++)
    {
        sampleCounter = 1;
        ret = CC_TST_TRNG(g_CcBaseAddr, TRNGMode, roscLength, sampleCounter,
FIRST_ITERATION_BUF_LEN, callback_FirstIteration);
        printData(ret, roscLength, sampleCounter);
        buffer_index = 0;
        memset(data_buf, 0, sizeof(data_buf));

        sampleCounter = 2;
        ret = CC_TST_TRNG(g_CcBaseAddr, TRNGMode, roscLength, sampleCounter,
FIRST_ITERATION_BUF_LEN, callback_FirstIteration);
        printData(ret, roscLength, sampleCounter);
        buffer_index = 0;
        memset(data_buf, 0, sizeof(data_buf));

        sampleCounter = 4;
        ret = CC_TST_TRNG(g_CcBaseAddr, TRNGMode, roscLength, sampleCounter,
FIRST_ITERATION_BUF_LEN, callback_FirstIteration);
        printData(ret, roscLength, sampleCounter);
        buffer_index = 0;
        memset(data_buf, 0, sizeof(data_buf));
    }
}
```

3.6 Second characterization iteration

You must run a second set of characterization tests to determine the TRNG configuration parameters.

Procedure

1. Collect data for each of the worst corners identified as result of the first iteration. Call `CC_TST_TRNG` with each of the four ring oscillator lengths (each with its corresponding sample count).

Depending on the intended TRNG driver, each call to `CC_TST_TRNG` must be as follows:

- FE TRNG driver: Call `CC_TST_TRNG` with `TRNGMode=1` and collect 100Mbit (12.5MB).
 - Arm recommends that all output bits be collected using a single contiguous execution of the test if your system allows. However, if there is insufficient memory to collect all required bits in a single run, you can split each test into several runs and concatenate the data. Each run must be a multiple of 1Mbits. Examples are splitting the test into 100 runs and collecting 100 x 1Mbits, 50 runs (50 x 2Mbits), or 25 runs (25 x 4Mbits).

- All resulting bits must be saved in the output file without any gaps.
- 800-90B TRNG driver: CC_TST_TRNG with *TRNGMode=2* and collect 1Mbit.
 - Due to the required size of the data, the output bits can be non-contiguous, and concatenation of several smaller sets of consecutive samples (generated using the same noise source) is allowed.
 - Smaller sets must contain at least 1000 samples.
 - The concatenated dataset must contain at least 1,000,000 samples.
 - All resulting bits must be saved in the output file without any gaps.



If any bits are dropped and not captured in the output file, you must rerun the test as the statistical analysis of the output is meaningless.

The same output file naming rules apply for both drivers. For more information, see [3.3.1 Output-file names](#) on page 13.

2. Process the resulting output files as follows:

- If you ran the characterization in AIS-31 mode, then you must send the resulting output files to Arm for statistical analysis.
- If you ran the characterization in 800-90 mode, then you must proceed and analyze the results using NIST tools, as published in the NIST site.

In either case, the returned results confirm or refute the TRNG configuration that is used for mass production. After the [3.7 Restart tests iteration](#) on page 19, these configuration values must be updated in the relevant TRNG driver files.



If there are errors, recollect the data for each failed corner with a larger sample count value. Then rerun the test and repeat until successful.

Example 3-3: Sample code of the second characterization test that is based on the FE TRNG driver

The following sample code includes the macro and callback function definitions.

```
/*
 * Define the buffer size for the second characterization iteration.
 * For example, at least 100 Mbits (12.5 MBytes) must be collected
 * every time EHR registers output 192 bits (24Bytes) value.
 * Therefore, set the collected data as 12500016 bytes (12500016/24 = 520834).
 * The full buffer size is 12500016+28(header+footer)=12500044 bytes.
 */
#define SECOND_ITERATION_BUFF_LEN      (12500044)

static void callback_SecondIteration(uint32_t outputSize, uint8_t *outputBuffer)
{
    int i = 0;
    if (NULL == outputBuffer)
    {

```

```
    printf("invalid input parameter!\r\n");  
}  
for (i=0; i<outputSize; i++)  
{  
    printf("%02x", *(outputBuffer+i));  
}  
}
```

Example 3-3: Second characterization test sample code

The following sample code is based on *callback_SecondIteration*.

```
void second_iteration_test()  
{  
    uint32_t TRNGMode = 1;  
    uint32_t roscLength = 0;  
    uint32_t sampleCounter = 1;  
    int ret = 0;  
    uint32_t i = 0;  
  
    for (roscLength=0; roscLength<4; roscLength++)  
    {  
        /*  
        * The following line must be updated for your board.  
        * It is based on Musca-B1 test boards.  
        */  
        if ((roscLength == 1) || (roscLength == 3))  
            continue;  
        /*  
        * The sample counter values in the following switch statement must  
        * be updated according the real cases. The following are the values  
        * for the tested Musca-B1 board.  
        */  
        switch (roscLength)  
        {  
            case 0:  
                sampleCounter = 360;  
                break;  
            case 1:  
                sampleCounter = 420;  
                break;  
            case 2:  
                sampleCounter = 600;  
                break;  
            case 3:  
                sampleCounter = 620;  
                break;  
        }  
        printf("\r\nroscLength=%d, sampleCount=%d\r\n", roscLength, sampleCounter);  
        ret = CC_TST_TRNG(g_CcBaseAddr, TRNGMode, roscLength, sampleCounter,  
SECOND_ITERATION_BUFF_LEN, callback_SecondIteration);  
        printf("\r\nret= %d\r\n", ret);  
    }  
}
```

3.7 Restart tests iteration

If you have selected the 800-90B driver, you must run a third characterization step to verify the results from the second iteration.

Before you begin

- Ensure that the sample counter values you use are from a successful second characterization iteration.

Procedure

1. Perform 1000 power cycles of the device, and in each cycle call `CC_TST_TRNG` with `TRNGMode = 0`. Collect 1000 bits under standard temperature, voltage, and process corner for each ring oscillator setting.
2. Concatenate the 1000 sequences one by one to form a single file for each ring oscillator.
3. Analyze the file, using the NIST tools, to confirm or refute the TRNG configuration that was calculated in the second iteration.
The NIST tools are published on the NIST site.

You must ensure that each file contains exactly 1,000,000 bits of collected samples. Because `CC_TST_TRNG` can output only multiples of 24 bytes (the length of the EHR register), this requirement is achieved by stripping the excess bits. Collect at least 1000 bits in each power cycle and keep only the first 1000 bits before concatenating the sequences together.

If there are failed tests, increase the sample counter value and repeat the restart tests.

Next steps

After verification, update these configuration values in the relevant files of the TRNG driver that you are using.

Appendix A CC_TST_TRNG output format

This appendix describes the format of the CC_TST_TRNG output.

When CC_TST_TRNG is run, in addition to collecting samples, it stores some metadata in its output buffer. This buffer is described in terms of 32-bit little-endian words.

The following table lists the value of each word.

Table A-1: CC_TST_TRNG output format

| Buffer offset (32-bit words) | Value |
|------------------------------|--|
| 0 | Signature value: 0xAABBCCDD. |
| 1 | <ul style="list-style-type: none"> Bits [23:0] - <i>buffSize</i>. Bits [25:24] - <i>roscLength</i>. Bits [31:30] - <i>TRNGMode</i>. |
| 2 | <i>sampleCount</i> |
| 3 | Signature value: 0xAABBCCDD. |
| 4..N-1 | Collected samples. Each 32-bit word contains the first collected sample in bit 0, and the last collected sample in bit 31. |
| N | Signature value: 0xDDCCBBAA. |
| N+1 | <p>Error flags</p> <ul style="list-style-type: none"> Bit [0] - Samples were lost during collection. Bit [1] - Autocorrelation error occurred. Bit [2] - CRNGT error that is detected and recovered. Bit [3] - Input that is stuck at same level for 32 bits. <p>Note: Bits[3:1] have no effect on the characterization analysis process.</p> |
| N+2 | Signature value: 0xDDCCBBAA. |

The value of "N" is derived from the buffer size, and is calculated to fit the entire data in the supplied buffer (as per the *buffSize* argument).

When parsing the output, it is important to test that the signature value is correct.

Appendix B Revision history

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

Table B-1: Issue 00 (v1.0)

| Change | Location |
|----------------|----------|
| First release. | - |

Table B-2: Differences between issue 00 (v1.0) and issue 01 (v1.1)

| Change | Location |
|---|---|
| Rebranded template to Arm logo and colors. | Entire document. |
| Product renamed Arm® TrustZone® TRNG. | Entire document. |
| <i>DX_TST_TRNG</i> renamed <i>CC_TST_TRNG</i> . | Entire document. |
| Added the following standards: <ul style="list-style-type: none"> BSI AIS-31: <i>Functionality Classes and Evaluation Methodology for True Random Number Generators</i> NIST SP 800-90A <i>Recommendation for Random Number Generation Using Deterministic Random Bit Generators – App C</i>. | Referenced Documents |
| Added STRNG driver operating mode. | <i>ArmTrustZone TRNG Overview</i> |
| Added STRNG driver to <i>TRNGMode</i> values in <i>CC_TST_TRNG</i> API code block. | First Characterization Test Program . |
| Added STRNG driver. | Second Characterization Test Program |
| Added STRNG driver to <i>TRNGMode</i> values in the <i>Reading Arm TrustZone TRNG Configuration Parameters</i> code block. | <i>Arm TrustZone TRNG Configuration Parameters</i> |

Table B-3: Differences between issue 01 (v1.1) and issue 02 (v1.2)

| Change | Location |
|----------------------------|------------------------------------|
| Renamed TRNG driver modes. | Entire document |
| Minor rephrasing. | <i>Arm TrustZone TRNG Overview</i> |

Table B-4: Differences between issue 02 (v1.2) and issue 03 (v1.3)

| Change | Location |
|---|---|
| Minor correction. | Referenced Documents |
| Rewritten. | Introduction |
| Added chapter. | 2 Overview of Arm True Random Number Generator (TRNG) on page 8 |
| Renamed from <i>ARM TrustZone TRNG Overview</i> (added under <i>Overview</i>) and partially rewritten. | <i>Arm TrustZone TRNG</i> |
| Renamed from <i>Characterization Overview</i> , and partially rewritten. | RNG Characterization |
| Merged <i>Characterization High-Level Procedure</i> into it and rewritten. | 3 TRNG characterization procedure on page 10 |
| Renamed from <i>Setting the Output Files</i> and partially rewritten. | 3.3.1 Output-file names on page 13 |
| Added the section. | 3.2 Characterization test program on page 11 |
| Rewritten. | 3.3 Characterization test conditions on page 12 |

| Change | Location |
|--|--|
| Added the section, including the <i>Finding Minimum Sample Counter Value</i> code block moved from <i>First Characterization Iteration</i> . | 3.4 Base iteration on page 13 |
| Removed the section, and moved all its subsections under <i>Characterization Procedure</i> . | <i>Characterization Detailed Procedure</i> |
| Renamed from <i>First Characterization Test Program</i> , replaced first paragraph and removed step 1; partially rewritten. | 3.5 First characterization iteration on page 15 |
| Renamed from <i>Second Characterization Test Program</i> , and rewritten. | 3.6 Second characterization iteration on page 17 |
| Removed the section. | <i>Arm TrustZone TRNG Configuration Parameters</i> |

Table B-5: Differences between issue 03 (v1.3) and issue 04 (v1.4)

| Change | Location |
|--|----------|
| Renamed from <i>Introduction</i> and restructured. | Preface |

Table B-6: Differences between issue 04 (v1.4) and issue 05

| Change | Location |
|--|-----------------|
| Template changes for the current releases. | Entire document |

Table B-7: Differences between issue 05 and issue 06

| Change | Location |
|--|--|
| Document reclassified as Non-Confidential. | Entire document |
| Minor rephrasing in multiple sections. No technical changes. | Entire document |
| Split content into a new 3.1 Characterization procedure overview on page 10 section. No technical changes. | 3 TRNG characterization procedure on page 10 |
| Split content into a new A CC_TST_TRNG output format on page 21 section. No technical changes. | A CC_TST_TRNG output format on page 21 |

Table B-8: Differences between issue 06 and issue 07

| Change | Location |
|---|---|
| <i>Compliance</i> section added. | 2.2 Compliance on page 8 |
| Additional information provided in <i>Overview</i> . | 2 Overview of Arm True Random Number Generator (TRNG) on page 8 |
| <ul style="list-style-type: none"> Note added before the table. Restart tests iteration added as an extra step. | 3.1 Characterization procedure overview on page 10 |
| Changes made to the procedure. | 3.6 Second characterization iteration on page 17 |
| <i>Restart tests iteration</i> section added. | 3.7 Restart tests iteration on page 19 |

Table B-9: Differences between issue 07 and issue 08

| Change | Location |
|--|--|
| Updated execution of "Second iteration: Run second set of characterization tests." | 3.1 Characterization procedure overview on page 10 |
| Changes made to the expected outcome of the procedure. | 3.5 First characterization iteration on page 15 |

Table B-10: Differences between issue 08 and issue 09

| Change | Location |
|-------------------------------------|-----------------|
| Removed TrustZone from product name | Entire document |

| Change | Location |
|----------|--|
| Updated. | 3.2 Characterization test program on page 11 |

Table B-11: Differences between issue 09 and issue 10

| Change | Location |
|---|--|
| Clarified that "Restart tests iteration" only applies to 800-90B mode. | 3.1 Characterization procedure overview on page 10 |
| <ul style="list-style-type: none"> Added paragraph re. location of the CC_TST_TRNGAPI. Updated API code example. Added definition of callback_TRNG code example and corresponding note. | 3.2 Characterization test program on page 11 |
| Updated collection conditions for each TRNG driver. | 3.3 Characterization test conditions on page 12 |
| <ul style="list-style-type: none"> Defined the amount of data that must be collected for the base iteration and the first characterization iteration. Added the Base iteration sample code on page 14 example code. Replaced the finding minimum sample counter value example code with Base iteration sample code on page 14. | 3.4 Base iteration on page 13 |
| <ul style="list-style-type: none"> Added sample code and updated first characterization iteration description. Added steps to be taken after the first characterization iteration, and before the second characterization iteration. | 3.5 First characterization iteration on page 15 |
| Added sample codes. | 3.6 Second characterization iteration on page 17 |
| Updated prerequisites: <ul style="list-style-type: none"> Only applies to 800-90B. <i>TRNGMode</i> must be set to 0. | 3.7 Restart tests iteration on page 19 |
| Buffer offset 1, updated <i>TRNGMode</i> bits from [31:31] to [31:30]. | A CC_TST_TRNG output format on page 21 |

Table B-12: Differences between issue 10 and issue 11

| Change | Location |
|--|--|
| Product renamed Arm® True Random Number Generator (TRNG). | Entire document |
| Clarified information about <i>printData</i> in example of sample code for first characterization iteration. | 3.5 First characterization iteration on page 15 |
| Corrected description of second characterization iteration procedure, that each test run must be a multiple of 1Mbits. | 3.6 Second characterization iteration on page 17 |
| Fixed buffer size values in the sample code of the second characterization test. | 3.6 Second characterization iteration on page 17 |
| Clarified description of restart tests iteration. Added extra information to step 3 of procedure. | 3.7 Restart tests iteration on page 19 |
| Reorganized appendix structure. No technical changes. | A CC_TST_TRNG output format on page 21 |