

# AN11643

## LPC18S/43S00, LPC18/4300 Random Number Generator

Rev. 1.0 — 03 February 2015

Application note

### Document information

Info	Content
<b>Keywords</b>	LPC1800,LPC4300, LPC18S00, LPC43S00, TRNG, True Random number generator, NIST, Diehard, RNG
<b>Abstract</b>	This application note describes the various steps to verify the randomness of numbers generated by the hardware random number generator present in the LPC18S00/LPC1800 and LPC43S00/LPC4300 devices.





## Revision history

Rev	Date	Description
1.0	20150203	Initial version

## Contact information

For additional information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)



## 1. Introduction

---

Random number generators are used for cryptographic, modeling and simulation applications which employ keys that must be generated in a random fashion. This application note focuses on the randomness required in cryptographic applications. The random number generator produces a stream of 0's and 1's with unpredictability. There are two types of random number generators: Pseudorandom Number Generator (PRNG) and True Random Number Generator (TRNG).

PRNG are deterministic random number generators which produce values from an input called the seed. The values produced by PRNG are predictable if the seed and generation algorithm are known. In many cases the generation algorithm is publicly available and therefore, to ensure unpredictability, the seed must be random and kept a secret. The seed itself must be obtained from the output of a TRNG.

TRNG are non-deterministic random number generators that use an unpredictable source (i.e. the entropy source) to produce randomness. The entropy source consists of a physical quantity such as noise of an electrical circuit or quantum effects of a semiconductor.

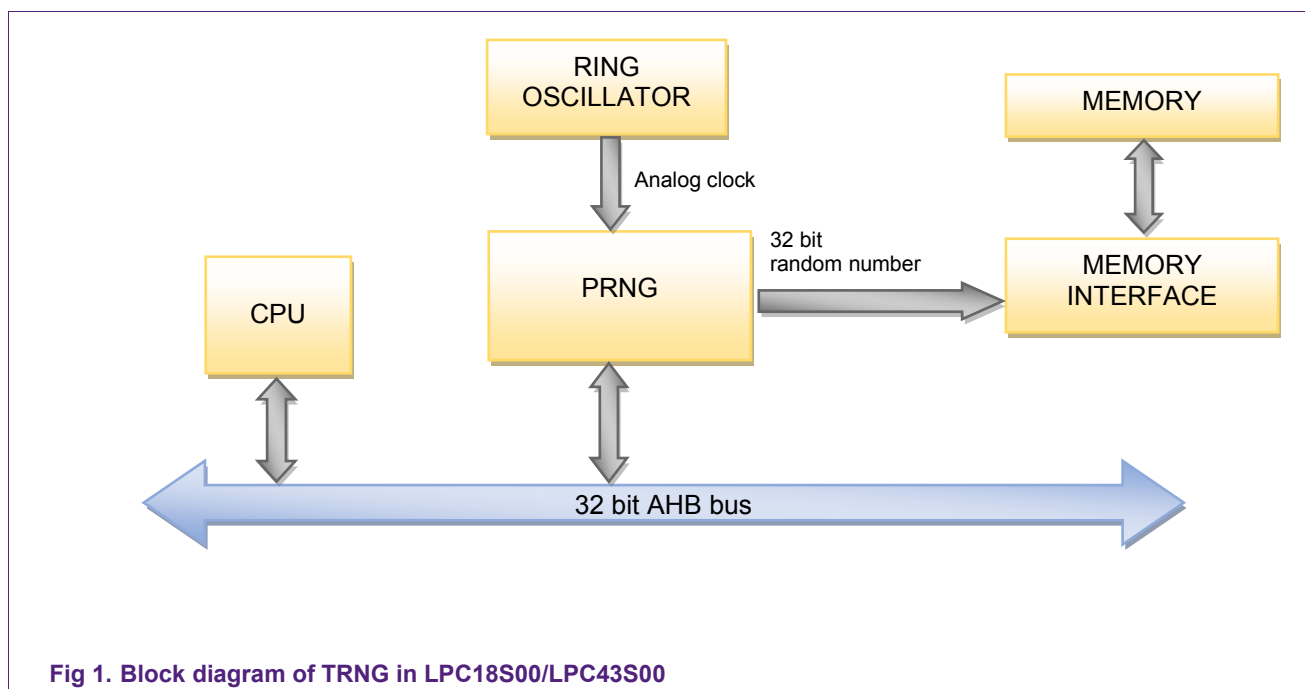
LPC1800 and LPC4300 contain a True Random Number Generator (TRNG). The output values of any RNG needs to satisfy the strict randomness criteria measured by statistical test tools in order to determine that the physical source of the RNG input is random.

## 2. True Random Number Generator (TRNG)

---

The main function of the TRNG is to provide a random number on request. The TRNG implemented in the LPC18S/43S00 and LPC18/4300 uses analog clocks as input. The analog clocks are derived from several ring oscillators. These analog clocks are fed to a pseudo random number generator to produce a 32 bit random number. Four 32 bit random words are concatenated by the bootrom to produce the 128 bit random number. Fig.1 shows the block diagram of the TRNG block.





### 3. Generation of Random numbers using TRNG block

An API is implemented in LPC18S/43S00 and LPC18/4300 devices which when invoked will generate a 128 bit random number. The *otp\_init()* API is used to initialize the OTP controller and the *Otp\_GenRand()* API is used to generate a random number from the hardware random number generator each time the API is invoked.

The 128-bit random number can be read from the following addresses:

- Bits 31:0 at location 0x4004 5050
- Bits 63:32 at location 0x4004 5054
- Bits 95:64 at location 0x4004 5058
- Bits 127:96 at location 0x4004 505C

The Keil MCB1800/4300 evaluation board with LPC1857/LPC4357 mounted on it as shown in Fig.2 is used in this application note.

The Evaluation board (OM #13040) can be ordered from the following link:

<http://www.nxp.com/demoboard/OM13040.html>

To set up the board:

- Connect a serial cable from UART3 on the Keil board to the PC. Make sure the jumpers TX(J16) and RX(J13) are on UART3.
- Connect a USB cable to power the board.
- Connect a suitable debugger – LPCLink2, Ulink2, Jlink etc.





**Fig 2. Keil MCB1800/4300 Evaluation Board**

The application note contains demonstration software for the following IDE's LPCXpresso, Keil MDK and IAR Workbench along with NIST and DIEHARD test suites.

The demonstration software generates 800,000 128-bit random numbers which is streamed to a PC terminal as binary sequence (ASCII characters consisting of 0's and 1's) using UART3 at 9600 baud rate. This constitutes to 102,400,000 bits. In this case TeraTerm was used as the terminal console and the random numbers are saved in a data file 'test\_data.dat. To log the data in TeraTerm, click File -> Log. Other terminals can of course be used that support similar features.

On the Keil board LED P9\_1 remains OFF when the random numbers are generated. After generating all the random numbers a "COMPLETED" message can be seen on the terminal and the LED P9\_1 will turn ON. This output file is analyzed by the NIST test suite to check for randomness. See next section for more details.

Note: Data input may be supplied in one of two ways. Files should contain binary sequences stored as either ASCII characters consisting of zeros and ones, or as binary data where each byte contains eight bits worth of 0's and 1's.



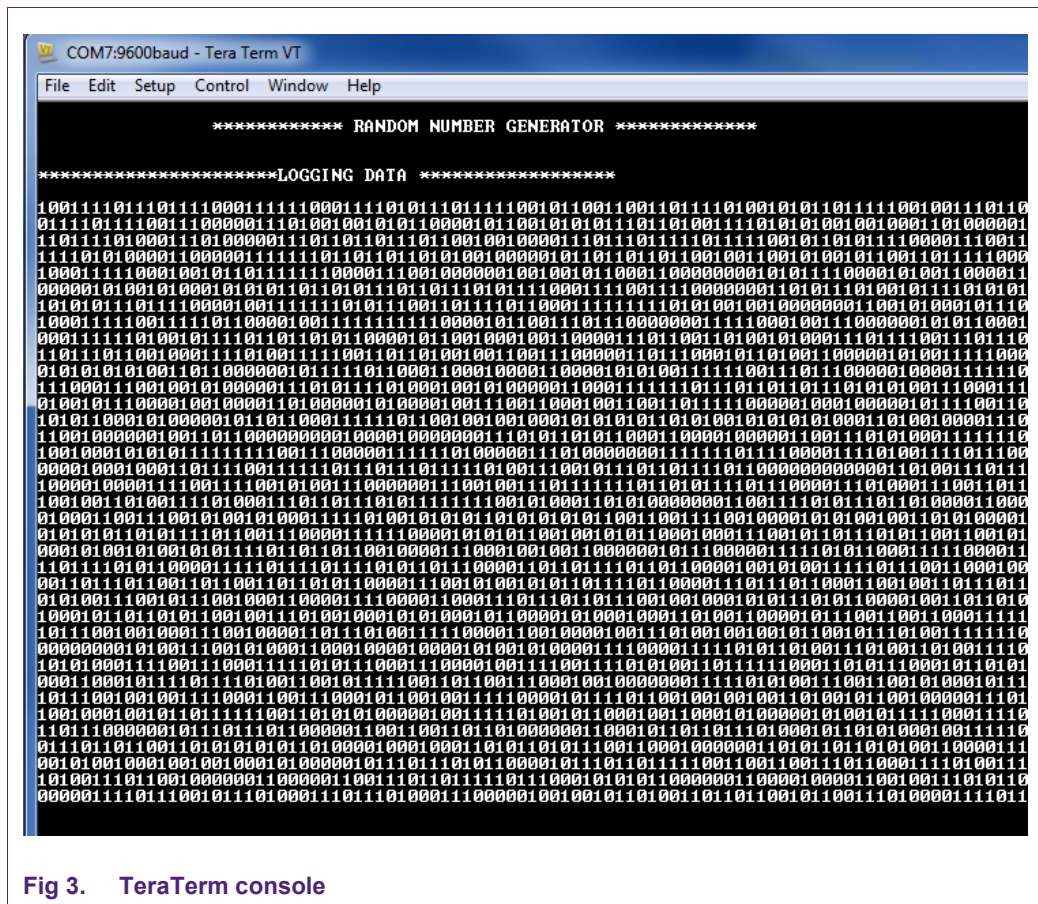


Fig 3. TeraTerm console

## 4. NIST SP800-22 test suite

### 4.1 Introduction

The National Institute of Standards and Technology (NIST) have developed a test suite which is a statistical tool to test the randomness of binary sequences of random number generators for cryptographic applications. A detailed description of the test suite can be found in the paper 'A Statistical Test Suite for Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications'. The paper and the test suite software package can be downloaded from the following link:

[http://csrc.nist.gov/groups/ST/toolkit/rng/documentation\\_software.html](http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html)

### 4.2 NIST SP800-22 test suite description

The NIST test suite consists of 15 tests to test the randomness in binary sequences. These tests focus on a variety of different types of non-randomness that could exist in a sequence. The various tests are listed below:

#### 4.2.1 Frequency (Monobit) Test

The purpose of this test is to determine if the distribution of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence.



#### 4.2.2 Frequency test within a Block

The purpose of this test is to determine if the frequency of ones in a block of  $M$  bits is approximately  $M/2$  as would be expected under the assumption of randomness.

#### 4.2.3 Runs test

The purpose of the runs test is to determine if the number of runs (uninterrupted sequence of identical bits) of ones and zeros of various lengths is as expected for a random sequence. A run of length  $k$  consists of exactly  $k$  identical bits and is bounded before and after with a bit of opposite value.

#### 4.2.4 Test for the Longest run of Ones in a Block

The purpose of this test is to determine if the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence.

#### 4.2.5 Binary Matrix Rank Test

The purpose of this test is to check for linear dependence among fixed length of substrings in the original sequence.

#### 4.2.6 Linear Complexity Test

The purpose of this test is to determine if a sequence is complex enough to be considered random. This is evaluated by the length of the Linear Feedback Shift Register (LFSR).

#### 4.2.7 Discrete Fourier Transform (Spectral) Test

The purpose of this test is to detect periodic features (i.e. repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness. The peak heights in the Discrete Fourier Transform of the sequence is given importance in this test.

#### 4.2.8 Non-overlapping Template Matching Test

The purpose of this test is to detect the number of occurrences of a specific  $m$ -bit aperiodic pattern.

#### 4.2.9 Overlapping Template Matching Test

The purpose of this test is to detect the number of occurrences of a specific  $m$ -bit periodic pattern.

#### 4.2.10 Maurer's "Universal Statistical" Test

The purpose of this test is to detect if a sequence can be compressed significantly without loss of information. A sequence is considered to be non-random if it can be significantly compressed.

#### 4.2.11 Serial Test

The purpose of this test is to determine if the number of occurrences of all possible  $m$ -bit overlapping patterns is approximately the same as would be expected for a random sequence.

#### 4.2.12 Approximate Entropy Test

The purpose of this test is to compare the frequency of the overlapping blocks of all  $m$ -bit with all  $m+1$  bit patterns.



#### 4.2.13 Cumulative Sums (Cusum) Test

The purpose of this test is to determine if the cumulative sum of the partial sequence is too small or too small compared to the expected behavior of that cumulative sum for random sequences.

#### 4.2.14 Random Excursions Test

The purpose of this test is to determine if the number of visits to a particular state within a cycle of random walk deviates from what would be expected for a random sequence.

#### 4.2.15 Random Excursions Variant Test

The purpose of this test is to detect the deviations from expected number of visits to various states in the random walk.

## 5. Analyzing the NIST SP800-22 Test Suite

### 5.1 Running NIST Test Suite

The NIST test suite package *sts-2.1.1* is used to verify the randomness of the output file generated by the LPC18S/43S00 and LPC18/4300 random number generator.

The generated file to be tested for randomness should be placed in the **data** folder found in the *sts-2.1.1* package (*sts-2.1.1\data*). Then the *sts-2.1.1* package is compiled as described in the NIST statistical test suite documentation to create an executable file “*assess.exe*” in the project directory. A gcc compiler can be used for this purpose. A compiled version of the test suite is provided with the application note.

#### 5.1.1 Step 1

In order to invoke the NIST statistical test suite, type **assess** followed by the desired bit stream length as shown in Fig.4. The output file to be analyzed consists of 100,000,000 bits with 10,000,000 bits per stream.

Enter choice ‘0’ in *GENERATOR SELECTION* menu to input the file to be analyzed.

```
C:\RNG\sts-2.1.1\sts-2.1.1>assess 100000000
      G E N E R A T O R   S E L E C T I O N

[0] Input File           [11] Linear Congruential
[2] Quadratic Congruential I  [9] Quadratic Congruential II
[4] Cubic Congruential      [5] XOR
[6] Modular Exponentiation  [7] Blum-Blum-Shub
[8] Micali-Schnorr         [9] G Using SHA-1

Enter Choice: 0
```

Fig 4. Generator selection



### 5.1.2 Step 2

Provide the path of the input file in the *User Prescribed Input File* field. The following screen is displayed and the statistical tests to be applied must be selected. In this case, '1' has been selected to apply all the tests.

```
User Prescribed Input File: data\test_data.dat

S T A T I S T I C A L   T E S T S

[01] Frequency                      [02] Block Frequency
[03] Cumulative Sums                [04] Runs
[05] Longest Run of Ones            [06] Rank
[07] Discrete Fourier Transform     [08] Nonperiodic Template Matchings
[09] Overlapping Template Matchings [10] Universal Statistical
[11] Approximate Entropy             [12] Random Excursions
[13] Random Excursions Variant       [14] Serial
[15] Linear Complexity

INSTRUCTIONS
Enter 0 if you DO NOT want to apply all of the
statistical tests to each sequence and 1 if you DO.

Enter Choice: 1
```

Fig 5. Statistical Tests

### 5.1.3 Step 3

The default settings are maintained and '0' is entered to continue. A query for the desired number of bit streams is then made. Here 10 sequences are parsed using the "test\_data.dat" file i.e. 10 blocks of 10,000,000 bits are selected which equals 100,000,000 bits.

```
Parameter Adjustments
-----
[1] Block Frequency Test - block length(M): 128
[2] NonOverlapping Template Test - block length(m): 9
[3] Overlapping Template Test - block length(m): 9
[4] Approximate Entropy Test - block length(m): 10
[5] Serial Test - block length(m): 16
[6] Linear Complexity Test - block length(M): 500

Select Test (0 to continue): 0

How many bitstreams? 10
```

Fig 6. Parametric adjustment and bitstreams input



#### 5.1.4 Step 4

In the following screen, the type of input file is specified. Value '0' is entered as the file is in ASCII format with sequence of 0's and 1's.

```
Input File Format:
[0] ASCII - A sequence of ASCII 0's and 1's
[1] Binary - Each byte in data file contains 8 bits of data
Select input mode: 0
```

Fig 7. Input file format

#### 5.1.5 Step 5

The test suite then proceeds to analyze the data. Once the test is complete the final results can be found in ***experiments\Algorithm Testing\FinalAnalysisReport.txt***.

```
Statistical Testing In Progress.....
Statistical Testing Complete!!!!!!!!!!!!!!
```

Fig 8. Statistical test In-Progress/Complete

#### 5.1.6 Test Report

The P-values in the test result refers to the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested, given the kind of non-randomness assessed by the test. In simple words if a P-value for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. A P-value of zero indicates that the sequence appears to be completely non-random.

Before performing the test a threshold value is chosen, called the significance level of the test denoted by  $\alpha$ . If  $P\text{-value} \geq \alpha$ , then the sequence appears to be random.

If  $P\text{-value} < \alpha$ , then the sequence appears to be non-random. Typically,  $\alpha$  is chosen in the range [0.001, 0.01].

For example if  $\alpha$  is 0.01, then one sequence in 100 sequences is random.

A  $P\text{-value} \geq 0.01$  would mean that the sequence would be considered to be random with a confidence of 99%. A  $P\text{-value} < 0.01$  would mean that the conclusion was that the sequence is non-random with a confidence of 99%. Refer to 'A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications' for more details on how P-values are calculated for each test.

The test results are represented in the form of a table. The rows indicate the tests performed and columns 1-10 indicate the frequency of p value, column 11 indicates the



P-value, column 12 indicates the proportion of binary sequences that passed the test and column 13 indicates the tests performed.

----- RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES -----												
generator is <data\test_data.dat>												
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
1	3	0	2	0	1	1	0	1	1	0.534146	10/10	Frequency
0	1	2	3	1	1	0	2	0	0	0.350485	10/10	BlockFrequency
1	2	1	2	1	1	1	0	0	1	0.911413	10/10	CumulativeSums
1	3	1	1	0	1	1	2	0	0	0.534146	10/10	CumulativeSums
1	0	0	1	0	2	2	0	1	3	0.350485	10/10	Runs
1	1	0	1	1	2	1	1	1	1	0.991468	10/10	LongestRun
1	2	2	1	1	0	1	2	0	0	0.739918	9/10	Rank
2	0	1	1	1	1	1	1	1	1	0.991468	10/10	FFT
0	1	1	0	1	1	2	1	1	2	0.911413	10/10	NonOverlappingTemplate
0	2	1	1	0	3	1	0	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	2	1	0	0	3	0	3	0.122325	10/10	NonOverlappingTemplate
1	1	1	1	0	1	2	0	2	1	0.911413	10/10	NonOverlappingTemplate
2	0	0	2	1	0	2	1	2	0	0.534146	10/10	NonOverlappingTemplate

**Fig 9. Depiction of Final Analysis Report**

The test result includes the interpretation guidelines to determine if the RNG passed the tests or failed. In this case, the minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 8 for a sample size = 10 binary sequences. The minimum pass rate for the random excursion (variant) test is approximately = 8 for a sample size = 9 binary sequences. This means that every single test can be considered as "passed" if it reports "X/10" where "X" is at least 8. The exception is the "RandomExcursions" tests, which report "Y/9" and the threshold is 8. Thus the RNG in LPC18S/43S00 and LPC18/4300 passes the NIST test suite. Refer to Appendix A for complete test report.



## 6. DIEHARD: A Battery of Tests of Randomness

The DIEHARD tests are a battery of statistical tests developed by George Marsaglia for measuring the quality of a random number generator. It consists of fifteen tests namely:

- 1) The birthday spacing's test
- 2) Overlapping 5-permutation test
- 3) The binary rank test for 31x31 matrices, 32x32 matrices and 6x8 matrices
- 4) The bitstream test
- 5) Overlapping Pairs Sparse Occupancy(OPSO)
- 6) Overlapping quadruples Sparse occupancy(OQSO)
- 7) DNA test
- 8) Count the 1's test
- 9) Parking lot test
- 10) Minimum distance test
- 11) 3Dspheres test
- 12) Squeeze test
- 13) Overlapping sums test
- 14) Runs test
- 15) Craps test

DIEHARD tests return a p-value. The 'p' value is uniform on [0, 1) if the input file contains independent random bits. The p-values are obtained by  $p=F(X)$ , where F is the assumed distribution of the sample random variable X. F is just an asymptotic approximation, for which the fit will be the worst in the tails. Thus occasionally p-values occur near 0 or 1, such as .0012 or .9983.

When a bit stream really fails, the tests result in p's of 0 or 1 in six or more places. One must not assume that a  $p < .025$  or  $p > .975$  means that the RNG has "failed the test at the .05 level". Such p's happen among the hundreds that DIEHARD produces, even with good RNG's. So it is important to note that "p happens". For a good RNG the 'p' value is uniformly distributed over [0, 1) and does not occur near 0 or 1.

For more information on the DIEHARD test suite and software please visit:

<http://stat.fsu.edu/pub/diehard/>

In this application note the windows version of DIEHARD software package is used for testing.

### 6.1 Running the DIEHARD Test Suite

#### 6.1.1 Step 1

An executable file "*DIEHARD.EXE*" is present in the diehard folder. When the program is run enter the input file to be tested and the output file to be generated. The output file is a log containing the results of the diehard test suite.

Here the input file is a binary file 'test\_data.bin' and the output file is 'test\_res.txt'.



```

C:\RNG\diehard\diehard>DIEHARD.EXE
NOTE: Most of the tests in DIEHARD return a p-value, which
should be uniform on [0,1) if the input file contains truly
independent random bits. Those p-values are obtained by
p=F(X), where F is the assumed distribution of the sample
random variable X---often normal. But that assumed F is just
an asymptotic approximation, for which the fit will be worst
in the tails. Thus you should not be surprised with
occasional p-values near 0 or 1, such as .0012 or .9983.
When a bit stream really FAILS BIG, you will get p's of 0 or
1 to six or more places. By all means, do not, as a
Statistician might, think that a p < .025 or p > .975 means
that the RNG has "failed the test at the .05 level". Such
p's happen among the hundreds that DIEHARD produces, even
with good RNG's. So keep in mind that "p happens".
Enter filename (<=15 characters):
test_data.bin
Enter name of output file (<=15 characters):
test_res.txt

```

Fig 10. Executable program

### 6.1.2 Step 2

Enter the tests that need to be performed. Here all the tests have been selected to be performed as indicated by fifteen 1's "111111111111111".

```

Which tests do you want performed?
For all tests, enter 15 1's:
111111111111111
For, say, tests 1,3,7 and 14, enter
101000100000010
HERE ARE YOUR CHOICES:
1 Birthday Spacings
2 Overlapping Permutations
3 Ranks of 31x31 and 32x32 matrices
4 Ranks of 6x8 Matrices
5 Monkey Tests on 20-bit Words
6 Monkey Tests OPS0,QQS0,DNA
7 Count the 1's in a Stream of Bytes
8 Count the 1's in Specific Bytes
9 Parking Lot Test
10 Minimum Distance Test
11 Random Spheres Test
12 The Squeeze Test
13 Overlapping Sums Test
14 Runs Test
15 The Craps Test
Enter your choices, 1's yes, 0's no. using 15 columns:
123456789012345
111111111111111

```

Fig 11. Choice of Tests



### 6.1.3 Test report

After all the tests are selected, the diehard test suite begins its testing process and once completed sends the test results to the output file 'test\_res.txt'. Fig. 12 shows an excerpt from the final test report. The entire test report can be found in Appendix A.

```
SUMMARY    FOR test_data.bin
           p-value for no. of wins : .422241
           p-value for throws/game : .586316

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Results of DIEHARD battery of tests sent to file test res.txt
```

**Fig 12. Test report**

For the performed tests:

- Number of p values  $\leq 0.025$  and p values  $\geq 0.975$  is 12.
- Number of p values  $\leq 0.01$  and p values  $\geq 0.99$  is 6.
- Number of p values  $\leq 0.0001$  and p values  $\geq 0.9999$  is 0.

## 7. Conclusion

The random number generator in LPC18S/43S00 and LPC18/4300 is a true random number generator that uses analog clocks as input which are unpredictable producing randomness. The RNG passes the NIST SP800-22 Rev.1a and DIEHARD test suite.

An important application of the random number generator is to generate 128 bit keys used in AES encryption and decryption operations in LPC18S/43S00 devices.



## 8. Appendix A

### 8.1 Test Report for NIST SP800-22 Test Suite

-----  
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES  
-----

generator is <data\test\_data.dat>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
1	3	0	2	0	1	1	0	1	1	0.534146	10/10	Frequency
0	1	2	3	1	1	0	2	0	0	0.350485	10/10	BlockFrequency
1	2	1	2	1	1	1	0	0	1	0.911413	10/10	CumulativeSums
1	3	1	1	0	1	1	2	0	0	0.534146	10/10	CumulativeSums
1	0	0	1	0	2	2	0	1	3	0.350485	10/10	Runs
1	1	0	1	1	2	1	1	1	1	0.991468	10/10	LongestRun
1	2	2	1	1	0	1	2	0	0	0.739918	9/10	Rank
2	0	1	1	1	1	1	1	1	1	0.991468	10/10	FFT
0	1	1	0	1	1	2	1	1	2	0.911413	10/10	NonOverlappingTemplate
0	2	1	1	0	3	1	0	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	2	1	0	0	3	0	3	0.122325	10/10	NonOverlappingTemplate
1	1	1	1	0	1	2	0	2	1	0.911413	10/10	NonOverlappingTemplate
2	0	0	2	1	0	2	1	2	0	0.534146	10/10	NonOverlappingTemplate
2	0	2	0	1	2	1	0	2	0	0.534146	10/10	NonOverlappingTemplate
0	1	1	0	3	0	1	2	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	1	1	1	2	2	1	1	0.911413	10/10	NonOverlappingTemplate
1	4	0	0	1	1	1	0	2	0	0.122325	10/10	NonOverlappingTemplate
0	1	0	2	1	3	1	1	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	1	1	0	1	1	3	1	0	0.739918	10/10	NonOverlappingTemplate
1	1	1	1	2	0	1	2	1	0	0.911413	10/10	NonOverlappingTemplate
1	1	1	2	1	0	1	1	2	0	0.911413	10/10	NonOverlappingTemplate
1	0	0	1	3	1	0	2	1	1	0.534146	10/10	NonOverlappingTemplate
4	2	0	0	1	2	1	0	0	0	0.066882	10/10	NonOverlappingTemplate
1	1	1	0	0	0	0	1	3	3	0.213309	10/10	NonOverlappingTemplate



## LPC18S/43S00, LPC18/4300 Random Number Generator

0	1	3	2	2	0	0	1	0	1	0.350485	10/10	NonOverlappingTemplate
2	0	0	2	0	0	2	0	3	1	0.213309	10/10	NonOverlappingTemplate
1	0	0	0	1	0	3	3	0	2	0.122325	9/10	NonOverlappingTemplate
2	1	2	0	1	1	1	2	0	0	0.739918	10/10	NonOverlappingTemplate
0	1	1	0	1	2	1	3	1	0	0.534146	10/10	NonOverlappingTemplate
2	1	3	0	1	2	0	0	1	0	0.350485	10/10	NonOverlappingTemplate
1	1	1	1	0	1	1	1	0	3	0.739918	10/10	NonOverlappingTemplate
0	0	0	1	2	3	2	0	0	2	0.213309	10/10	NonOverlappingTemplate
0	0	1	1	1	1	0	1	3	2	0.534146	10/10	NonOverlappingTemplate
1	1	0	1	3	2	0	0	2	0	0.350485	10/10	NonOverlappingTemplate
1	1	1	1	2	1	1	0	1	1	0.991468	10/10	NonOverlappingTemplate
1	2	0	2	0	1	3	0	0	1	0.350485	9/10	NonOverlappingTemplate
1	0	1	2	0	1	0	3	0	2	0.350485	10/10	NonOverlappingTemplate
0	2	0	2	2	1	1	0	2	0	0.534146	10/10	NonOverlappingTemplate
0	0	0	2	2	1	2	1	1	1	0.739918	10/10	NonOverlappingTemplate
1	3	0	2	0	2	1	1	0	0	0.350485	10/10	NonOverlappingTemplate
2	2	1	1	0	1	0	0	1	2	0.739918	10/10	NonOverlappingTemplate
0	1	2	1	1	0	2	1	2	0	0.739918	10/10	NonOverlappingTemplate
2	1	2	2	1	1	1	0	0	0	0.739918	10/10	NonOverlappingTemplate
0	0	2	0	0	0	3	4	1	0	0.017912	10/10	NonOverlappingTemplate
3	3	0	1	1	0	1	1	0	0	0.213309	9/10	NonOverlappingTemplate
1	0	2	0	1	1	3	1	0	1	0.534146	9/10	NonOverlappingTemplate
1	0	2	2	0	0	1	1	1	2	0.739918	10/10	NonOverlappingTemplate
2	1	1	1	0	0	2	0	2	1	0.739918	10/10	NonOverlappingTemplate
2	0	1	1	0	2	0	2	1	1	0.739918	10/10	NonOverlappingTemplate
1	1	0	0	3	1	0	0	3	1	0.213309	10/10	NonOverlappingTemplate
2	1	0	0	1	0	1	2	1	2	0.739918	10/10	NonOverlappingTemplate
0	1	2	0	0	2	0	1	1	3	0.350485	10/10	NonOverlappingTemplate
1	0	0	0	1	1	1	1	1	4	0.213309	10/10	NonOverlappingTemplate
1	3	1	1	1	0	3	0	0	0	0.213309	10/10	NonOverlappingTemplate
0	1	1	2	0	0	0	2	2	2	0.534146	10/10	NonOverlappingTemplate
1	0	0	1	1	2	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
1	1	1	2	0	1	0	3	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	0	1	3	2	0	0	2	0	0.350485	10/10	NonOverlappingTemplate
1	1	4	1	0	0	2	0	1	0	0.122325	9/10	NonOverlappingTemplate
2	1	2	2	1	1	1	0	0	0	0.739918	10/10	NonOverlappingTemplate
0	1	1	2	2	1	3	0	0	0	0.350485	10/10	NonOverlappingTemplate



1	1	1	1	1	2	0	0	3	0	0.534146	9/10	NonOverlappingTemplate
5	0	1	1	1	1	0	0	1	0	0.017912	10/10	NonOverlappingTemplate
1	1	3	2	1	0	0	0	1	1	0.534146	10/10	NonOverlappingTemplate
0	1	1	3	1	1	0	0	2	1	0.534146	10/10	NonOverlappingTemplate
2	1	0	0	1	1	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
3	0	0	0	3	0	1	0	1	2	0.122325	9/10	NonOverlappingTemplate
0	0	2	3	0	0	2	1	2	0	0.213309	10/10	NonOverlappingTemplate
1	0	1	1	2	1	2	0	2	0	0.739918	10/10	NonOverlappingTemplate
2	1	2	0	0	0	1	1	2	1	0.739918	10/10	NonOverlappingTemplate
1	1	0	1	1	2	2	1	1	0	0.911413	10/10	NonOverlappingTemplate
1	0	0	0	1	2	2	1	2	1	0.739918	10/10	NonOverlappingTemplate
2	1	0	1	0	0	2	1	1	2	0.739918	10/10	NonOverlappingTemplate
1	0	1	1	2	2	3	0	0	0	0.350485	10/10	NonOverlappingTemplate
0	2	1	0	3	1	1	2	0	0	0.350485	10/10	NonOverlappingTemplate
0	2	0	3	0	0	0	2	2	1	0.213309	10/10	NonOverlappingTemplate
1	1	2	1	2	0	1	1	0	1	0.911413	10/10	NonOverlappingTemplate
1	0	0	2	3	0	1	1	1	1	0.534146	10/10	NonOverlappingTemplate
1	2	1	0	0	1	0	1	2	2	0.739918	10/10	NonOverlappingTemplate
2	0	0	2	2	3	0	0	1	0	0.213309	10/10	NonOverlappingTemplate
1	0	1	0	1	1	1	1	1	3	0.739918	10/10	NonOverlappingTemplate
1	0	1	2	0	1	3	1	1	0	0.534146	10/10	NonOverlappingTemplate
0	1	1	0	1	1	2	1	1	2	0.911413	10/10	NonOverlappingTemplate
2	0	2	1	0	2	1	2	0	0	0.534146	9/10	NonOverlappingTemplate
3	1	0	3	0	2	1	0	0	0	0.122325	9/10	NonOverlappingTemplate
0	0	2	1	1	0	2	0	1	3	0.350485	10/10	NonOverlappingTemplate
1	0	2	0	1	1	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
1	2	0	0	2	3	0	1	0	1	0.350485	10/10	NonOverlappingTemplate
2	1	1	3	0	0	0	1	2	0	0.350485	9/10	NonOverlappingTemplate
0	0	2	0	0	1	2	2	1	2	0.534146	10/10	NonOverlappingTemplate
0	1	1	1	2	1	0	1	3	0	0.534146	10/10	NonOverlappingTemplate
0	2	1	1	1	1	2	1	0	1	0.911413	10/10	NonOverlappingTemplate
1	2	2	0	2	0	0	1	1	1	0.739918	10/10	NonOverlappingTemplate
3	0	0	0	1	2	2	0	0	2	0.213309	10/10	NonOverlappingTemplate
1	2	1	0	0	3	0	1	0	2	0.350485	10/10	NonOverlappingTemplate
0	0	0	4	1	2	2	0	0	1	0.066882	10/10	NonOverlappingTemplate
0	2	1	1	2	1	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
1	1	1	1	0	2	0	3	0	1	0.534146	10/10	NonOverlappingTemplate



1	1	3	1	1	1	0	2	0	0	0.534146	10/10	NonOverlappingTemplate
1	0	2	2	2	0	0	2	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	1	0	3	1	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
0	1	1	0	1	2	1	0	3	1	0.534146	10/10	NonOverlappingTemplate
1	2	0	1	0	2	0	1	2	1	0.739918	10/10	NonOverlappingTemplate
3	0	1	0	1	1	1	1	1	1	0.739918	10/10	NonOverlappingTemplate
1	0	1	0	2	3	0	2	0	1	0.350485	10/10	NonOverlappingTemplate
1	0	2	3	0	1	1	0	1	1	0.534146	9/10	NonOverlappingTemplate
0	3	1	1	0	2	2	0	0	1	0.350485	10/10	NonOverlappingTemplate
1	1	2	2	2	1	0	0	1	0	0.739918	10/10	NonOverlappingTemplate
1	1	2	0	1	0	3	1	1	0	0.534146	9/10	NonOverlappingTemplate
2	0	1	3	1	1	2	0	0	0	0.350485	10/10	NonOverlappingTemplate
0	3	1	0	0	1	1	3	0	1	0.213309	10/10	NonOverlappingTemplate
0	0	2	2	1	2	2	1	0	0	0.534146	10/10	NonOverlappingTemplate
1	2	0	1	2	1	2	1	0	0	0.739918	10/10	NonOverlappingTemplate
0	1	1	1	1	0	1	2	2	1	0.911413	10/10	NonOverlappingTemplate
1	0	2	0	3	3	0	0	1	0	0.122325	10/10	NonOverlappingTemplate
0	0	0	1	0	0	2	2	2	3	0.213309	10/10	NonOverlappingTemplate
0	1	0	3	1	0	2	2	1	0	0.350485	10/10	NonOverlappingTemplate
2	1	0	0	0	1	2	2	1	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	2	0	0	2	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	3	1	0	0	1	0	1	2	2	0.350485	10/10	NonOverlappingTemplate
4	0	0	0	1	0	3	1	0	1	0.035174	10/10	NonOverlappingTemplate
2	0	0	1	2	0	0	3	2	0	0.213309	10/10	NonOverlappingTemplate
3	2	0	1	3	0	0	0	1	0	0.122325	10/10	NonOverlappingTemplate
2	0	0	0	0	2	3	1	2	0	0.213309	10/10	NonOverlappingTemplate
0	0	1	2	0	2	0	1	2	2	0.534146	10/10	NonOverlappingTemplate
1	1	1	1	1	1	1	2	0	1	0.991468	10/10	NonOverlappingTemplate
0	1	2	0	1	0	1	1	1	3	0.534146	10/10	NonOverlappingTemplate
2	0	1	2	1	0	1	0	1	2	0.739918	10/10	NonOverlappingTemplate
1	4	1	0	0	1	0	1	2	0	0.122325	10/10	NonOverlappingTemplate
2	0	1	0	0	3	1	1	1	1	0.534146	10/10	NonOverlappingTemplate
1	0	1	0	2	2	2	1	0	1	0.739918	10/10	NonOverlappingTemplate
0	2	1	2	2	0	0	0	2	1	0.534146	10/10	NonOverlappingTemplate
1	0	0	1	1	2	0	1	3	1	0.534146	10/10	NonOverlappingTemplate
0	0	0	0	3	2	2	1	1	1	0.350485	10/10	NonOverlappingTemplate
1	2	0	1	1	3	1	0	1	0	0.534146	10/10	NonOverlappingTemplate



## LPC18S/43S00, LPC18/4300 Random Number Generator

1	2	0	2	0	1	1	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	2	0	1	1	2	3	0	1	0	0.350485	10/10	NonOverlappingTemplate
1	0	1	0	0	2	2	2	2	0	0.534146	10/10	NonOverlappingTemplate
2	0	2	0	4	1	0	0	1	0	0.066882	10/10	NonOverlappingTemplate
0	1	0	1	1	0	1	1	2	3	0.534146	10/10	NonOverlappingTemplate
1	0	1	3	0	0	2	2	1	0	0.350485	10/10	NonOverlappingTemplate
0	0	1	1	0	3	0	1	0	4	0.035174	10/10	NonOverlappingTemplate
0	1	2	1	1	1	1	0	2	1	0.911413	10/10	NonOverlappingTemplate
0	0	0	3	2	1	2	0	1	1	0.350485	10/10	NonOverlappingTemplate
0	1	2	0	2	1	2	1	0	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	2	1	0	0	0	4	0	0.122325	10/10	NonOverlappingTemplate
0	2	2	1	0	1	1	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	0	2	0	4	0	2	0	1	1	0.066882	10/10	NonOverlappingTemplate
1	4	0	1	0	0	0	2	0	2	0.066882	10/10	NonOverlappingTemplate
0	2	0	1	2	1	1	1	1	1	0.911413	10/10	NonOverlappingTemplate
1	0	1	0	1	0	1	3	2	1	0.534146	10/10	NonOverlappingTemplate
0	1	6	1	0	1	1	0	0	0	0.000439	10/10	NonOverlappingTemplate
1	0	0	0	1	1	3	3	1	0	0.213309	10/10	NonOverlappingTemplate
1	1	0	0	0	1	1	1	4	1	0.213309	10/10	NonOverlappingTemplate
1	1	2	1	0	0	1	1	1	2	0.911413	10/10	NonOverlappingTemplate
1	0	1	2	0	1	3	1	1	0	0.534146	10/10	NonOverlappingTemplate
3	0	2	2	0	1	1	0	1	0	0.350485	10/10	OverlappingTemplate
1	2	3	0	0	0	2	0	1	1	0.350485	10/10	Universal
0	1	2	1	2	1	1	2	0	0	0.739918	10/10	ApproximateEntropy
0	0	1	3	1	1	0	1	1	1	----	9/9	RandomExcursions
1	1	0	0	0	1	2	1	0	3	----	9/9	RandomExcursions
0	1	1	1	1	0	2	1	1	1	----	9/9	RandomExcursions
1	0	0	0	4	2	1	0	1	0	----	9/9	RandomExcursions
0	1	1	0	1	0	1	0	3	2	----	9/9	RandomExcursions
0	2	1	2	0	2	0	1	1	0	----	9/9	RandomExcursions
2	0	1	0	1	0	1	2	0	2	----	9/9	RandomExcursions
3	0	1	1	0	2	1	0	0	1	----	9/9	RandomExcursions
0	0	1	0	3	1	1	1	1	1	----	9/9	RandomExcursionsVariant
0	0	1	1	2	1	0	1	3	0	----	9/9	RandomExcursionsVariant
0	0	0	1	2	2	3	0	0	1	----	9/9	RandomExcursionsVariant
0	0	1	0	4	2	0	0	0	2	----	9/9	RandomExcursionsVariant
0	0	2	1	0	1	2	3	0	0	----	9/9	RandomExcursionsVariant



0	2	0	1	0	0	2	1	2	1	----	9/9	RandomExcursionsVariant
1	1	0	1	2	2	0	1	0	1	----	9/9	RandomExcursionsVariant
2	0	0	1	1	0	2	3	0	0	----	9/9	RandomExcursionsVariant
1	1	0	0	0	3	0	2	1	1	----	9/9	RandomExcursionsVariant
2	1	0	0	0	2	0	1	1	2	----	9/9	RandomExcursionsVariant
2	1	2	0	0	0	1	0	1	2	----	9/9	RandomExcursionsVariant
4	0	0	0	1	0	1	1	2	0	----	9/9	RandomExcursionsVariant
4	0	0	0	0	2	1	0	2	0	----	9/9	RandomExcursionsVariant
3	1	0	0	0	2	1	1	1	0	----	8/9	RandomExcursionsVariant
2	1	0	2	0	0	1	0	2	1	----	9/9	RandomExcursionsVariant
2	1	0	2	0	1	0	0	1	2	----	9/9	RandomExcursionsVariant
2	1	1	2	0	1	0	0	2	0	----	9/9	RandomExcursionsVariant
2	1	2	1	0	1	1	0	0	1	----	9/9	RandomExcursionsVariant
2	1	0	2	3	0	0	0	1	1	0.350485	10/10	Serial
2	2	1	1	0	1	0	1	0	2	0.739918	9/10	Serial
2	0	0	2	3	1	1	0	1	0	0.350485	10/10	LinearComplexity

-----

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 8 for a sample size = 10 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately = 8 for a sample size = 9 binary sequences.

-----



## 8.2 Test Report for DIEHARD Test Suite

NOTE: Most of the tests in DIEHARD return a p-value, which should be uniform on [0,1) if the input file contains truly independent random bits. Those p-values are obtained by  $p=F(X)$ , where  $F$  is the assumed distribution of the sample random variable  $X$ —often normal. But that assumed  $F$  is just an asymptotic approximation, for which the fit will be worst in the tails. Thus you should not be surprised with occasional p-values near 0 or 1, such as .0012 or .9983. When a bit stream really FAILS BIG, you will get p's of 0 or 1 to six or more places. By all means, do not, as a Statistician might, think that a  $p < .025$  or  $p > .975$  means that the RNG has "failed the test at the .05 level". Such p's happen among the hundreds that DIEHARD produces, even with good RNG's. So keep in mind that "p happens".

```

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::          This is the BIRTHDAY SPACINGS TEST          ::
:: Choose m birthdays in a year of n days. List the spacings ::
:: between the birthdays. If j is the number of values that ::
:: occur more than once in that list, then j is asymptotically ::
:: Poisson distributed with mean  $m^2/(4n)$ . Experience shows n ::
:: must be quite large, say  $n \geq 2^{18}$ , for comparing the results ::
:: to the Poisson distribution with that mean. This test uses ::
::  $n=2^{24}$  and  $m=2^9$ , so that the underlying distribution for j ::
:: is taken to be Poisson with  $\lambda=2^{27}/(2^{26})=2$ . A sample ::
:: of 500 j's is taken, and a chi-square goodness of fit test ::
:: provides a p value. The first test uses bits 1-24 (counting ::
:: from the left) from integers in the specified file.      ::
:: Then the file is closed and reopened. Next, bits 2-25 are ::
:: used to provide birthdays, then 3-26 and so on to bits 9-32. ::
:: Each set of bits provides a p-value, and the nine p-values ::
:: provide a sample for a KSTEST.                          ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
BIRTHDAY SPACINGS TEST, M= 512 N=2**24 LAMBDA= 2.0000
Results for test_data.bin
For a sample of size 500:      mean
test_data.bin using bits 1 to 24 2.060
duplicate      number      number
spacings      observed     expected
0             72.         67.668
1             129.        135.335
2             124.        135.335
3             95.         90.224
4             48.         45.112
5             22.         18.045
6 to INF      10.         8.282
Chisquare with 6 d.o.f. =      3.18 p-value= .214651
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

```



```

                For a sample of size 500:      mean
      test_data.bin  using bits  2 to 25  2.024
duplicate      number      number
spacings      observed    expected
    0          71.         67.668
    1         120.         135.335
    2         143.         135.335
    3          91.         90.224
    4          51.         45.112
    5          19.         18.045
  6 to INF         5.         8.282
Chisquare with  6 d.o.f. =      4.46 p-value= .385611
::::::::::::::::::::::::::::::::::::::::

```

```

                For a sample of size 500:      mean
      test_data.bin  using bits  3 to 26  2.098
duplicate      number      number
spacings      observed    expected
    0          64.         67.668
    1         125.         135.335
    2         133.         135.335
    3          95.         90.224
    4          54.         45.112
    5          19.         18.045
  6 to INF        10.         8.282
Chisquare with  6 d.o.f. =      3.44 p-value= .248000
::::::::::::::::::::::::::::::::::::::::

```

```

                For a sample of size 500:      mean
      test_data.bin  using bits  4 to 27  1.974
duplicate      number      number
spacings      observed    expected
    0          65.         67.668
    1         136.         135.335
    2         144.         135.335
    3          86.         90.224
    4          50.         45.112
    5          12.         18.045
  6 to INF         7.         8.282
Chisquare with  6 d.o.f. =      3.61 p-value= .271234
::::::::::::::::::::::::::::::::::::::::

```

```

                For a sample of size 500:      mean
      test_data.bin  using bits  5 to 28  2.032
duplicate      number      number
spacings      observed    expected
    0          65.         67.668
    1         133.         135.335
    2         133.         135.335
    3          95.         90.224
    4          48.         45.112
    5          18.         18.045
  6 to INF         8.         8.282

```



Chisquare with 6 d.o.f. = .63 p-value= .004180

::

For a sample of size 500: mean

test\_data.bin using bits 6 to 29 2.038

duplicate	number	number
spacings	observed	expected
0	57.	67.668
1	131.	135.335
2	155.	135.335
3	88.	90.224
4	38.	45.112
5	26.	18.045
6 to INF	5.	8.282

Chisquare with 6 d.o.f. = 10.66 p-value= .900585

::

For a sample of size 500: mean

test\_data.bin using bits 7 to 30 2.010

duplicate	number	number
spacings	observed	expected
0	77.	67.668
1	115.	135.335
2	133.	135.335
3	104.	90.224
4	53.	45.112
5	11.	18.045
6 to INF	7.	8.282

Chisquare with 6 d.o.f. = 10.81 p-value= .905718

::

For a sample of size 500: mean

test\_data.bin using bits 8 to 31 1.918

duplicate	number	number
spacings	observed	expected
0	81.	67.668
1	131.	135.335
2	141.	135.335
3	77.	90.224
4	48.	45.112
5	13.	18.045
6 to INF	9.	8.282

Chisquare with 6 d.o.f. = 6.60 p-value= .640417

::

For a sample of size 500: mean

test\_data.bin using bits 9 to 32 2.000

duplicate	number	number
spacings	observed	expected
0	70.	67.668
1	131.	135.335
2	140.	135.335
3	86.	90.224
4	45.	45.112



```

      5      19.      18.045
6 to INF      9.      8.282
Chisquare with 6 d.o.f. =      .69 p-value=      .005314
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
The 9 p-values were
      .214651      .385611      .248000      .271234      .004180
      .900585      .905718      .640417      .005314
A KSTEST for the 9 p-values yields      .878112

```

\$

```

.....
::          THE OVERLAPPING 5-PERMUTATION TEST          ::
:: This is the OPERM5 test.  It looks at a sequence of one mill- ::
:: ion 32-bit random integers.  Each set of five consecutive ::
:: integers can be in one of 120 states, for the 5! possible ::
:: orderings of five numbers.  Thus the 5th, 6th, 7th,...numbers ::
:: each provide a state. As many thousands of state transitions ::
:: are observed, cumulative counts are made of the number of ::
:: occurrences of each state. Then the quadratic form in the ::
:: weak inverse of the 120x120 covariance matrix yields a test ::
:: equivalent to the likelihood ratio test that the 120 cell ::
:: counts came from the specified (asymptotically) normal dis- ::
:: tribution with the specified 120x120 covariance matrix (with ::
:: rank 99). This version uses 1,000,000 integers, twice.      ::
.....

```

```
OPERM5 test for file test_data.bin
```

For a sample of 1,000,000 consecutive 5-tuples,  
chisquare for 99 degrees of freedom=104.159; p-value= .658298

```
OPERM5 test for file test_data.bin
```

For a sample of 1,000,000 consecutive 5-tuples,  
chisquare for 99 degrees of freedom=101.574; p-value= .590452

```

.....
:: This is the BINARY RANK TEST for 31x31 matrices. The leftmost ::
:: 31 bits of 31 random integers from the test sequence are used ::
:: to form a 31x31 binary matrix over the field {0,1}. The rank ::
:: is determined. That rank can be from 0 to 31, but ranks< 28 ::
:: are rare, and their counts are pooled with those for rank 28. ::
:: Ranks are found for 40,000 such random matrices and a chisqua-::
:: re test is performed on counts for ranks 31,30,29 and <=28. ::
.....

```

Binary rank test for test\_data.bin

Rank test for 31x31 binary matrices:

rows from leftmost 31 bits of each 32-bit integer

rank	observed	expected	(o-e)^2/e	sum
28	232	211.4	2.003699	2.004
29	5153	5134.0	.070240	2.074
30	23171	23103.0	.199871	2.274
31	11444	11551.5	1.000864	3.275

chisquare= 3.275 for 3 d. of f.; p-value= .683995



```

-----
::: This is the BINARY RANK TEST for 32x32 matrices. A random 32x
::: 32 binary matrix is formed, each row a 32-bit random integer. ::
::: The rank is determined. That rank can be from 0 to 32, ranks ::
::: less than 29 are rare, and their counts are pooled with those ::
::: for rank 29. Ranks are found for 40,000 such random matrices ::
::: and a chisquare test is performed on counts for ranks 32,31, ::
::: 30 and <=29. ::
:::
Binary rank test for test_data.bin
Rank test for 32x32 binary matrices:
rows from leftmost 32 bits of each 32-bit integer
rank  observed  expected (o-e)^2/e  sum
29      209      211.4   .027655   .028
30     5156     5134.0   .094185   .122
31    23002    23103.0   .441953   .564
32    11633    11551.5   .574666   1.138
chisquare= 1.138 for 3 d. of f.; p-value= .375191
-----

```

\$

```

::: This is the BINARY RANK TEST for 6x8 matrices. From each of ::
::: six random 32-bit integers from the generator under test, a ::
::: specified byte is chosen, and the resulting six bytes form a ::
::: 6x8 binary matrix whose rank is determined. That rank can be ::
::: from 0 to 6, but ranks 0,1,2,3 are rare; their counts are ::
::: pooled with those for rank 4. Ranks are found for 100,000 ::
::: random matrices, and a chi-square test is performed on ::
::: counts for ranks 6,5 and <=4. ::
:::
Binary Rank Test for test_data.bin
Rank of a 6x8 binary matrix,
rows formed from eight bits of the RNG test_data.bin
b-rank test for bits 1 to 8
OBSERVED  EXPECTED  (O-E)^2/E  SUM
r<=4      934      944.3    .112      .112
r =5     21838     21743.9   .407      .520
r =6     77228     77311.8   .091      .610
p=1-exp(-SUM/2)= .26304
Rank of a 6x8 binary matrix,
rows formed from eight bits of the RNG test_data.bin
b-rank test for bits 2 to 9
OBSERVED  EXPECTED  (O-E)^2/E  SUM
r<=4      920      944.3    .625      .625
r =5     21671     21743.9   .244      .870
r =6     77409     77311.8   .122      .992
p=1-exp(-SUM/2)= .39104

```



Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 3 to 10

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r<=4	973	944.3	.872	.872
r =5	21832	21743.9	.357	1.229
r =6	77195	77311.8	.176	1.406

$p=1-\exp(-\text{SUM}/2)=.50481$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 4 to 11

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r<=4	948	944.3	.014	.014
r =5	21857	21743.9	.588	.603
r =6	77195	77311.8	.176	.779

$p=1-\exp(-\text{SUM}/2)=.32269$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 5 to 12

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r<=4	957	944.3	.171	.171
r =5	21735	21743.9	.004	.174
r =6	77308	77311.8	.000	.175

$p=1-\exp(-\text{SUM}/2)=.08360$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 6 to 13

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r<=4	963	944.3	.370	.370
r =5	21837	21743.9	.399	.769
r =6	77200	77311.8	.162	.931

$p=1-\exp(-\text{SUM}/2)=.37204$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 7 to 14

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r<=4	1024	944.3	6.727	6.727
r =5	21524	21743.9	2.224	8.950
r =6	77452	77311.8	.254	9.205

$p=1-\exp(-\text{SUM}/2)=.98997$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 8 to 15

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r<=4	960	944.3	.261	.261
r =5	21871	21743.9	.743	1.004
r =6	77169	77311.8	.264	1.268

$p=1-\exp(-\text{SUM}/2)=.46945$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin



b-rank test for bits 9 to 16

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r ≤ 4	972	944.3	.812	.812
r = 5	21727	21743.9	.013	.826
r = 6	77301	77311.8	.002	.827

$$p=1-\exp(-\text{SUM}/2)=.33871$$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin

b-rank test for bits 10 to 17

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r ≤ 4	935	944.3	.092	.092
r = 5	21645	21743.9	.450	.541
r = 6	77420	77311.8	.151	.693

$$p=1-\exp(-\text{SUM}/2)=.29280$$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin

b-rank test for bits 11 to 18

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r ≤ 4	932	944.3	.160	.160
r = 5	21739	21743.9	.001	.161
r = 6	77329	77311.8	.004	.165

$$p=1-\exp(-\text{SUM}/2)=.07927$$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin

b-rank test for bits 12 to 19

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r ≤ 4	937	944.3	.056	.056
r = 5	21504	21743.9	2.647	2.703
r = 6	77559	77311.8	.790	3.494

$$p=1-\exp(-\text{SUM}/2)=.82567$$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin

b-rank test for bits 13 to 20

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r ≤ 4	1011	944.3	4.711	4.711
r = 5	21715	21743.9	.038	4.750
r = 6	77274	77311.8	.018	4.768

$$p=1-\exp(-\text{SUM}/2)=.90782$$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin

b-rank test for bits 14 to 21

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
r ≤ 4	982	944.3	1.505	1.505
r = 5	21850	21743.9	.518	2.023
r = 6	77168	77311.8	.267	2.290

$$p=1-\exp(-\text{SUM}/2)=.68181$$

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin

b-rank test for bits 15 to 22

	OBSERVED	EXPECTED	(O-E) <sup>2</sup> /E	SUM
--	----------	----------	-----------------------	-----



r<=4	936	944.3	.073	.073
r =5	21861	21743.9	.631	.704
r =6	77203	77311.8	.153	.857
p=1-exp(-SUM/2)= .34843				

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 16 to 23

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	975	944.3	.998	.998
r =5	21571	21743.9	1.375	2.373
r =6	77454	77311.8	.262	2.634
p=1-exp(-SUM/2)= .73211				

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 17 to 24

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	969	944.3	.646	.646
r =5	21580	21743.9	1.235	1.881
r =6	77451	77311.8	.251	2.132
p=1-exp(-SUM/2)= .65563				

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 18 to 25

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	945	944.3	.001	.001
r =5	21874	21743.9	.778	.779
r =6	77181	77311.8	.221	1.000
p=1-exp(-SUM/2)= .39354				

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 19 to 26

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	953	944.3	.080	.080
r =5	21746	21743.9	.000	.080
r =6	77301	77311.8	.002	.082
p=1-exp(-SUM/2)= .04010				

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 20 to 27

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	982	944.3	1.505	1.505
r =5	21743	21743.9	.000	1.505
r =6	77275	77311.8	.018	1.523
p=1-exp(-SUM/2)= .53294				

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 21 to 28

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	966	944.3	.499	.499
r =5	21604	21743.9	.900	1.399



```

r =6      77430      77311.8      .181      1.579
p=1-exp(-SUM/2)= .54602

```

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 22 to 29

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	938	944.3	.042	.042
r =5	21795	21743.9	.120	.162
r =6	77267	77311.8	.026	.188

p=1-exp(-SUM/2)= .08976

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 23 to 30

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	936	944.3	.073	.073
r =5	21385	21743.9	5.924	5.997
r =6	77679	77311.8	1.744	7.741

p=1-exp(-SUM/2)= .97915

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 24 to 31

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	914	944.3	.972	.972
r =5	21522	21743.9	2.265	3.237
r =6	77564	77311.8	.823	4.060

p=1-exp(-SUM/2)= .86863

Rank of a 6x8 binary matrix,  
rows formed from eight bits of the RNG test\_data.bin  
b-rank test for bits 25 to 32

	OBSERVED	EXPECTED	(O-E)^2/E	SUM
r<=4	920	944.3	.625	.625
r =5	21535	21743.9	2.007	2.632
r =6	77545	77311.8	.703	3.336

p=1-exp(-SUM/2)= .81135

TEST SUMMARY, 25 tests on 100,000 random 6x8 matrices  
These should be 25 uniform [0,1] random variables:

.263041	.391035	.504809	.322686	.083598
.372044	.989972	.469455	.338708	.292796
.079269	.825674	.907820	.681811	.348427
.732113	.655628	.393544	.040096	.532937
.546025	.089763	.979151	.868634	.811352

brank test summary for test\_data.bin

The KS test for those 25 supposed UNI's yields  
KS p-value= .066557

\$

```

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::                                THE BITSTREAM TEST                                ::
:: The file under test is viewed as a stream of bits. Call them ::

```



```

:: b1,b2,... . Consider an alphabet with two "letters", 0 and 1 ::
:: and think of the stream of bits as a succession of 20-letter ::
:: "words", overlapping. Thus the first word is b1b2...b20, the ::
:: second is b2b3...b21, and so on. The bitstream test counts ::
:: the number of missing 20-letter (20-bit) words in a string of ::
:: 2^21 overlapping 20-letter words. There are 2^20 possible 20 ::
:: letter words. For a truly random string of 2^21+19 bits, the ::
:: number of missing words j should be (very close to) normally ::
:: distributed with mean 141,909 and sigma 428. Thus ::
:: (j-141909)/428 should be a standard normal variate (z score) ::
:: that leads to a uniform [0,1) p value. The test is repeated ::
:: twenty times. ::
:::

```

THE OVERLAPPING 20-tuples BITSTREAM TEST, 20 BITS PER WORD, N words

This test uses  $N=2^{21}$  and samples the bitstream 20 times.

No. missing words should average 141909. with  $\sigma=428$ .

```

-----
tst no 1: 141979 missing words,      .16 sigmas from mean, p-value= .56466
tst no 2: 141835 missing words,     -.17 sigmas from mean, p-value= .43106
tst no 3: 141828 missing words,     -.19 sigmas from mean, p-value= .42465
tst no 4: 141615 missing words,     -.69 sigmas from mean, p-value= .24583
tst no 5: 141363 missing words,    -1.28 sigmas from mean, p-value= .10090
tst no 6: 141777 missing words,     -.31 sigmas from mean, p-value= .37859
tst no 7: 141994 missing words,      .20 sigmas from mean, p-value= .57841
tst no 8: 141873 missing words,     -.08 sigmas from mean, p-value= .46618
tst no 9: 141815 missing words,     -.22 sigmas from mean, p-value= .41278
tst no 10: 141520 missing words,    -.91 sigmas from mean, p-value= .18150
tst no 11: 142063 missing words,     .36 sigmas from mean, p-value= .64022
tst no 12: 142096 missing words,     .44 sigmas from mean, p-value= .66864
tst no 13: 142170 missing words,     .61 sigmas from mean, p-value= .72875
tst no 14: 142167 missing words,     .60 sigmas from mean, p-value= .72643
tst no 15: 141585 missing words,    -.76 sigmas from mean, p-value= .22429
tst no 16: 142464 missing words,     1.30 sigmas from mean, p-value= .90251
tst no 17: 141739 missing words,     -.40 sigmas from mean, p-value= .34533
tst no 18: 142603 missing words,     1.62 sigmas from mean, p-value= .94746
tst no 19: 141567 missing words,     -.80 sigmas from mean, p-value= .21190
tst no 20: 142779 missing words,     2.03 sigmas from mean, p-value= .97892

```

\$

```

:::
::      The tests OPSO, OQSO and DNA ::
::      OPSO means Overlapping-Pairs-Sparse-Occupancy ::
:: The OPSO test considers 2-letter words from an alphabet of ::
:: 1024 letters. Each letter is determined by a specified ten ::
:: bits from a 32-bit integer in the sequence to be tested. OPSO ::
:: generates 2^21 (overlapping) 2-letter words (from 2^21+1 ::
:: "keystrokes") and counts the number of missing words---that ::
:: is 2-letter words which do not appear in the entire sequence. ::
:: That count should be very close to normally distributed with ::

```



```

:: mean 141,909, sigma 290. Thus (missingwrds-141909)/290 should ::
:: be a standard normal variable. The OPSO test takes 32 bits at ::
:: a time from the test file and uses a designated set of ten ::
:: consecutive bits. It then restarts the file for the next de- ::
:: signed 10 bits, and so on. ::
:: ::
:: QSO means Overlapping-Quadruples-Sparse-Occupancy ::
:: The test QSO is similar, except that it considers 4-letter ::
:: words from an alphabet of 32 letters, each letter determined ::
:: by a designated string of 5 consecutive bits from the test ::
:: file, elements of which are assumed 32-bit random integers. ::
:: The mean number of missing words in a sequence of 2^21 four- ::
:: letter words, (2^21+3 "keystrokes"), is again 141909, with ::
:: sigma = 295. The mean is based on theory; sigma comes from ::
:: extensive simulation. ::
:: ::
:: The DNA test considers an alphabet of 4 letters:: C,G,A,T,::
:: determined by two designated bits in the sequence of random ::
:: integers being tested. It considers 10-letter words, so that ::
:: as in OPSO and QSO, there are 2^20 possible words, and the ::
:: mean number of missing words from a string of 2^21 (over- ::
:: lapping) 10-letter words (2^21+9 "keystrokes") is 141909. ::
:: The standard deviation sigma=339 was determined as for QSO ::
:: by simulation. (Sigma for OPSO, 290, is the true value (to ::
:: three places), not determined by simulation. ::
:: ::
OPSO test for generator test_data.bin
Output: No. missing words (mw), equiv normal variate (z), p-value (p)

      mw      z      p
OPSO for test_data.bin using bits 23 to 32      141796  -.391  .3480
OPSO for test_data.bin using bits 22 to 31      141603 -1.056  .1454
OPSO for test_data.bin using bits 21 to 30      142316  1.402  .9196
OPSO for test_data.bin using bits 20 to 29      141555 -1.222  .1109
OPSO for test_data.bin using bits 19 to 28      142405  1.709  .9563
OPSO for test_data.bin using bits 18 to 27      142383  1.633  .9488
OPSO for test_data.bin using bits 17 to 26      141798  -.384  .3505
OPSO for test_data.bin using bits 16 to 25      141980  .244  .5963
OPSO for test_data.bin using bits 15 to 24      141837  -.249  .4015
OPSO for test_data.bin using bits 14 to 23      141733  -.608  .2716
OPSO for test_data.bin using bits 13 to 22      142262  1.216  .8880
OPSO for test_data.bin using bits 12 to 21      142023  .392  .6525
OPSO for test_data.bin using bits 11 to 20      141598 -1.074  .1415
OPSO for test_data.bin using bits 10 to 19      141584 -1.122  .1310
OPSO for test_data.bin using bits 9 to 18       141998  .306  .6201
OPSO for test_data.bin using bits 8 to 17       141941  .109  .5435
OPSO for test_data.bin using bits 7 to 16       141984  .257  .6016
OPSO for test_data.bin using bits 6 to 15       141667  -.836  .2017
OPSO for test_data.bin using bits 5 to 14       141322 -2.025  .0214
OPSO for test_data.bin using bits 4 to 13       141914  .016  .5064
OPSO for test_data.bin using bits 3 to 12       141601 -1.063  .1438

```



```

OPSO for test_data.bin using bits 2 to 11      142089  .620  .7322
OPSO for test_data.bin using bits 1 to 10      141662  -.853  .1969

```

QQSO test for generator test\_data.bin

Output: No. missing words (mw), equiv normal variate (z), p-value (p)

		mw	z	p
QQSO for test_data.bin	using bits 28 to 32	141895	-.049	.4806
QQSO for test_data.bin	using bits 27 to 31	142532	2.111	.9826
QQSO for test_data.bin	using bits 26 to 30	142261	1.192	.8834
QQSO for test_data.bin	using bits 25 to 29	140807	-3.737	.0001
QQSO for test_data.bin	using bits 24 to 28	141807	-.347	.3643
QQSO for test_data.bin	using bits 23 to 27	142253	1.165	.8780
QQSO for test_data.bin	using bits 22 to 26	141897	-.042	.4833
QQSO for test_data.bin	using bits 21 to 25	141876	-.113	.4550
QQSO for test_data.bin	using bits 20 to 24	141637	-.923	.1780
QQSO for test_data.bin	using bits 19 to 23	141587	-1.093	.1373
QQSO for test_data.bin	using bits 18 to 22	142373	1.572	.9420
QQSO for test_data.bin	using bits 17 to 21	142153	.826	.7956
QQSO for test_data.bin	using bits 16 to 20	141817	-.313	.3771
QQSO for test_data.bin	using bits 15 to 19	141772	-.466	.3208
QQSO for test_data.bin	using bits 14 to 18	141887	-.076	.4698
QQSO for test_data.bin	using bits 13 to 17	141337	-1.940	.0262
QQSO for test_data.bin	using bits 12 to 16	142178	.911	.8188
QQSO for test_data.bin	using bits 11 to 15	141559	-1.188	.1175
QQSO for test_data.bin	using bits 10 to 14	141663	-.835	.2019
QQSO for test_data.bin	using bits 9 to 13	141821	-.299	.3823
QQSO for test_data.bin	using bits 8 to 12	141684	-.764	.2225
QQSO for test_data.bin	using bits 7 to 11	141814	-.323	.3733
QQSO for test_data.bin	using bits 6 to 10	141708	-.682	.2475
QQSO for test_data.bin	using bits 5 to 9	141592	-1.076	.1410
QQSO for test_data.bin	using bits 4 to 8	142323	1.402	.9196
QQSO for test_data.bin	using bits 3 to 7	142416	1.718	.9571
QQSO for test_data.bin	using bits 2 to 6	142124	.728	.7666
QQSO for test_data.bin	using bits 1 to 5	142254	1.168	.8787

DNA test for generator test\_data.bin

Output: No. missing words (mw), equiv normal variate (z), p-value (p)

		mw	z	p
DNA for test_data.bin	using bits 31 to 32	141591	-.939	.1739
DNA for test_data.bin	using bits 30 to 31	141531	-1.116	.1322
DNA for test_data.bin	using bits 29 to 30	142378	1.383	.9166
DNA for test_data.bin	using bits 28 to 29	141661	-.733	.2319
DNA for test_data.bin	using bits 27 to 28	142028	.350	.6369
DNA for test_data.bin	using bits 26 to 27	141621	-.851	.1975
DNA for test_data.bin	using bits 25 to 26	142040	.385	.6501
DNA for test_data.bin	using bits 24 to 25	141873	-.107	.4573
DNA for test_data.bin	using bits 23 to 24	141807	-.302	.3814
DNA for test_data.bin	using bits 22 to 23	141535	-1.104	.1348
DNA for test_data.bin	using bits 21 to 22	141797	-.331	.3702
DNA for test_data.bin	using bits 20 to 21	141533	-1.110	.1335
DNA for test_data.bin	using bits 19 to 20	141618	-.859	.1951
DNA for test_data.bin	using bits 18 to 19	141928	.055	.5220



DNA for test_data.bin	using bits 17 to 18	141708	-.594	.2763
DNA for test_data.bin	using bits 16 to 17	142015	.312	.6224
DNA for test_data.bin	using bits 15 to 16	141923	.040	.5161
DNA for test_data.bin	using bits 14 to 15	141951	.123	.5489
DNA for test_data.bin	using bits 13 to 14	141712	-.582	.2803
DNA for test_data.bin	using bits 12 to 13	142420	1.506	.9340
DNA for test_data.bin	using bits 11 to 12	142528	1.825	.9660
DNA for test_data.bin	using bits 10 to 11	142510	1.772	.9618
DNA for test_data.bin	using bits 9 to 10	141444	-1.373	.0849
DNA for test_data.bin	using bits 8 to 9	141566	-1.013	.1556
DNA for test_data.bin	using bits 7 to 8	141845	-.190	.4247
DNA for test_data.bin	using bits 6 to 7	141368	-1.597	.0552
DNA for test_data.bin	using bits 5 to 6	141570	-1.001	.1584
DNA for test_data.bin	using bits 4 to 5	141970	.179	.5710
DNA for test_data.bin	using bits 3 to 4	141679	-.679	.2484
DNA for test_data.bin	using bits 2 to 3	142017	.318	.6246
DNA for test_data.bin	using bits 1 to 2	141553	-1.051	.1466

\$

```

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::      This is the COUNT-THE-1's TEST on a stream of bytes.      ::
:: Consider the file under test as a stream of bytes (four per    ::
:: 32 bit integer). Each byte can contain from 0 to 8 1's,        ::
:: with probabilities 1,8,28,56,70,56,28,8,1 over 256. Now let    ::
:: the stream of bytes provide a string of overlapping 5-letter   ::
:: words, each "letter" taking values A,B,C,D,E. The letters are ::
:: determined by the number of 1's in a byte:: 0,1,or 2 yield A,::
:: 3 yields B, 4 yields C, 5 yields D and 6,7 or 8 yield E. Thus ::
:: we have a monkey at a typewriter hitting five keys with vari- ::
:: ous probabilities (37,56,70,56,37 over 256). There are 5^5    ::
:: possible 5-letter words, and from a string of 256,000 (over-  ::
:: lapping) 5-letter words, counts are made on the frequencies   ::
:: for each word. The quadratic form in the weak inverse of     ::
:: the covariance matrix of the cell counts provides a chisquare ::
:: test:: Q5-Q4, the difference of the naive Pearson sums of    ::
:: (OBS-EXP)^2/EXP on counts for 5- and 4-letter cell counts.   ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

```

Test results for test\_data.bin

Chi-square with  $5^5 - 5^4 = 2500$  d.of f. for sample size:2560000

chisquare equiv normal p-value

Results fo COUNT-THE-1's in successive bytes:

byte stream for test_data.bin	2462.26	-.534	.296780
byte stream for test_data.bin	2502.64	.037	.514888

\$

```

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::      This is the COUNT-THE-1's TEST for specific bytes.      ::
:: Consider the file under test as a stream of 32-bit integers.  ::

```



```
:: From each integer, a specific byte is chosen , say the left- ::
:: most:: bits 1 to 8. Each byte can contain from 0 to 8 1's, ::
:: with probabilitie 1,8,28,56,70,56,28,8,1 over 256. Now let ::
:: the specified bytes from successive integers provide a string ::
:: of (overlapping) 5-letter words, each "letter" taking values ::
:: A,B,C,D,E. The letters are determined by the number of 1's, ::
:: in that byte:: 0,1,or 2 ---> A, 3 ---> B, 4 ---> C, 5 ---> D,::
:: and 6,7 or 8 ---> E. Thus we have a monkey at a typewriter ::
:: hitting five keys with with various probabilities:: 37,56,70,::
:: 56,37 over 256. There are 5^5 possible 5-letter words, and ::
:: from a string of 256,000 (overlapping) 5-letter words, counts ::
:: are made on the frequencies for each word. The quadratic form ::
:: in the weak inverse of the covariance matrix of the cell ::
:: counts provides a chisquare test:: Q5-Q4, the difference of ::
:: the naive Pearson sums of (OBS-EXP)^2/EXP on counts for 5- ::
:: and 4-letter cell counts. ::
:::
Chi-square with 5^5-5^4=2500 d.of f. for sample size: 256000
chisquare equiv normal p value
Results for COUNT-THE-1's in specified bytes:
bits 1 to 8 2423.22 -1.086 .138763
bits 2 to 9 2440.58 -.840 .200354
bits 3 to 10 2573.10 1.034 .849387
bits 4 to 11 2440.04 -.848 .198232
bits 5 to 12 2490.26 -.138 .445202
bits 6 to 13 2471.84 -.398 .345213
bits 7 to 14 2505.28 .075 .529776
bits 8 to 15 2505.52 .078 .531106
bits 9 to 16 2499.91 -.001 .499519
bits 10 to 17 2512.21 .173 .568547
bits 11 to 18 2564.08 .906 .817602
bits 12 to 19 2540.14 .568 .714865
bits 13 to 20 2512.74 .180 .571517
bits 14 to 21 2392.98 -1.513 .065084
bits 15 to 22 2495.12 -.069 .472472
bits 16 to 23 2581.68 1.155 .875977
bits 17 to 24 2519.66 .278 .609524
bits 18 to 25 2415.09 -1.201 .114903
bits 19 to 26 2457.69 -.598 .274790
bits 20 to 27 2510.17 .144 .557155
bits 21 to 28 2474.55 -.360 .359446
bits 22 to 29 2429.15 -1.002 .158192
bits 23 to 30 2489.51 -.148 .441055
bits 24 to 31 2594.37 1.335 .908986
bits 25 to 32 2478.23 -.308 .379105
```

\$

```
:::
:: THIS IS A PARKING LOT TEST ::
```



```

:: In a square of side 100, randomly "park" a car---a circle of  ::
:: radius 1. Then try to park a 2nd, a 3rd, and so on, each  ::
:: time parking "by ear". That is, if an attempt to park a car  ::
:: causes a crash with one already parked, try again at a new  ::
:: random location. (To avoid path problems, consider parking  ::
:: helicopters rather than cars.) Each attempt leads to either  ::
:: a crash or a success, the latter followed by an increment to  ::
:: the list of cars already parked. If we plot n: the number of  ::
:: attempts, versus k:: the number successfully parked, we get a  ::
:: curve that should be similar to those provided by a perfect  ::
:: random number generator. Theory for the behavior of such a  ::
:: random curve seems beyond reach, and as graphics displays are  ::
:: not available for this battery of tests, a simple characteriz  ::
:: ation of the random experiment is used: k, the number of cars  ::
:: successfully parked after n=12,000 attempts. Simulation shows  ::
:: that k should average 3523 with sigma 21.9 and is very close  ::
:: to normally distributed. Thus (k-3523)/21.9 should be a st-  ::
:: andard normal variable, which, converted to a uniform varia-  ::
:: ble, provides input to a KSTEST based on a sample of 10.  ::
:::

```

CDPARK: result of ten tests on file test\_data.bin

Of 12,000 tries, the average no. of successes

should be 3523 with sigma=21.9

Successes: 3549	z-score: 1.187	p-value: .882429
Successes: 3513	z-score: -.457	p-value: .323972
Successes: 3519	z-score: -.183	p-value: .427537
Successes: 3561	z-score: 1.735	p-value: .958644
Successes: 3511	z-score: -.548	p-value: .291865
Successes: 3524	z-score: .046	p-value: .518210
Successes: 3498	z-score: -1.142	p-value: .126820
Successes: 3510	z-score: -.594	p-value: .276387
Successes: 3568	z-score: 2.055	p-value: .980051
Successes: 3517	z-score: -.274	p-value: .392053

square size	avg. no. parked	sample sigma
100.	3527.000	22.517

KSTEST for the above 10: p= .472995

\$

```

:::
:: THE MINIMUM DISTANCE TEST ::
:: It does this 100 times:: choose n=8000 random points in a  ::
:: square of side 10000. Find d, the minimum distance between  ::
:: the (n^2-n)/2 pairs of points. If the points are truly inde-  ::
:: pendent uniform, then d^2, the square of the minimum distance  ::
:: should be (very close to) exponentially distributed with mean  ::
:: .995 . Thus 1-exp(-d^2/.995) should be uniform on [0,1) and  ::
:: a KSTEST on the resulting 100 values serves as a test of uni-  ::
:: formity for random points in the square. Test numbers=0 mod 5 ::

```



```
:: are printed but the KSTEST is based on the full set of 100    ::
:: random choices of 8000 points in the 10000x10000 square.    ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

This is the MINIMUM DISTANCE test

for random integers in the file test\_data.bin

sample no.	d^2	avg	equiv uni
5	2.6238	1.0687	.928422
10	.1233	1.3600	.116581
15	.5428	1.6158	.420438
20	.0789	1.4969	.076243
25	.5645	1.4936	.432952
30	.5591	1.4780	.429904
35	1.1406	1.4071	.682184
40	2.7788	1.3974	.938746
45	4.1706	1.4061	.984877
50	2.0574	1.3604	.873531
55	.0856	1.2767	.082455
60	2.0317	1.2385	.870225
65	.1478	1.2393	.138022
70	.7478	1.1966	.528392
75	.3255	1.1395	.279023
80	.2130	1.1015	.192677
85	.0449	1.0952	.044163
90	.4119	1.0845	.339003
95	.6400	1.0799	.474402
100	.2540	1.0598	.225326

MINIMUM DISTANCE TEST for test\_data.bin

Result of KS test on 20 transformed mindist^2's:

p-value= .389288

\$

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::          THE 3DSPHERES TEST          ::
:: Choose 4000 random points in a cube of edge 1000. At each ::
:: point, center a sphere large enough to reach the next closest ::
:: point. Then the volume of the smallest such sphere is (very ::
:: close to) exponentially distributed with mean 120pi/3. Thus ::
:: the radius cubed is exponential with mean 30. (The mean is ::
:: obtained by extensive simulation). The 3DSPHERES test gener- ::
:: ates 4000 such spheres 20 times. Each min radius cubed leads ::
:: to a uniform variable by means of 1-exp(-r^3/30.), then a ::
:: KSTEST is done on the 20 p-values. ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

The 3DSPHERES test for file test\_data.bin

sample no: 1	r^3=	61.128	p-value= .86966
sample no: 2	r^3=	36.022	p-value= .69903
sample no: 3	r^3=	13.859	p-value= .36995
sample no: 4	r^3=	74.907	p-value= .91766
sample no: 5	r^3=	28.472	p-value= .61290



```

sample no: 6      r^3= 24.528      p-value= .55851
sample no: 7      r^3=  3.017      p-value= .09568
sample no: 8      r^3=  6.725      p-value= .20081
sample no: 9      r^3= 15.352      p-value= .40054
sample no: 10     r^3= 50.286      p-value= .81292
sample no: 11     r^3=  .078      p-value= .00259
sample no: 12     r^3= 62.273      p-value= .87454
sample no: 13     r^3= 18.518      p-value= .46058
sample no: 14     r^3= 41.255      p-value= .74720
sample no: 15     r^3= 41.598      p-value= .75008
sample no: 16     r^3= 12.085      p-value= .33158
sample no: 17     r^3=  1.856      p-value= .06000
sample no: 18     r^3= 32.813      p-value= .66505
sample no: 19     r^3= 16.502      p-value= .42309
sample no: 20     r^3= 38.141      p-value= .71955
A KS test is applied to those 20 p-values.

```

```

-----
3DSPHERES test for file test_data.bin      p-value= .239388
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

```

```

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::      This is the SQUEEZE test      ::
:: Random integers are floated to get uniforms on [0,1). Start- ::
:: ing with k=2^31=2147483647, the test finds j, the number of ::
:: iterations necessary to reduce k to 1, using the reduction ::
:: k=ceiling(k*U), with U provided by floating integers from ::
:: the file being tested. Such j's are found 100,000 times, ::
:: then counts for the number of times j was <=6,7,...,47,>=48 ::
:: are used to provide a chi-square test for cell frequencies. ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

```

RESULTS OF SQUEEZE TEST FOR test\_data.bin

Table of standardized frequency counts

( (obs-exp)/sqrt(exp) )^2

for j taking values <=6,7,8,...,47,>=48:

-1.1	-.7	-.6	-.5	.5	-.4
.2	-1.2	-.3	.4	-.7	1.3
-.4	-1.1	.2	-.1	1.0	-1.6
1.8	-.9	-.7	.6	.6	2.2
-.8	-.4	.5	-1.1	-1.8	.0
.9	-.3	.9	.9	-.6	-.1
1.0	.8	-.8	.4	.1	-1.0
-1.1					

Chi-square with 42 degrees of freedom: 33.612

z-score= -.915 p-value= .181084

```

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

```

```

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::      The OVERLAPPING SUMS test      ::

```



```

:: Integers are floated to get a sequence U(1),U(2),... of uni- ::
:: form [0,1) variables. Then overlapping sums, ::
:: S(1)=U(1)+...+U(100), S2=U(2)+...+U(101),... are formed. ::
:: The S's are virtually normal with a certain covariance mat- ::
:: rix. A linear transformation of the S's converts them to a ::
:: sequence of independent standard normals, which are converted ::
:: to uniform variables for a KSTEST. The p-values from ten ::
:: KSTESTs are given still another KSTEST. ::
:::
Test no. 1 p-value .445004
Test no. 2 p-value .808746
Test no. 3 p-value .454994
Test no. 4 p-value .642899
Test no. 5 p-value .451406
Test no. 6 p-value .991862
Test no. 7 p-value .797197
Test no. 8 p-value .825439
Test no. 9 p-value .002200
Test no. 10 p-value .835099
Results of the OSUM test for test_data.bin
KSTEST on the above 10 p-values: .856181

```

\$

```

:::
:: This is the RUNS test. It counts runs up, and runs down, ::
:: in a sequence of uniform [0,1) variables, obtained by float- ::
:: ing the 32-bit integers in the specified file. This example ::
:: shows how runs are counted: .123,.357,.789,.425,.224,.416,.95::
:: contains an up-run of length 3, a down-run of length 2 and an ::
:: up-run of (at least) 2, depending on the next values. The ::
:: covariance matrices for the runs-up and runs-down are well ::
:: known, leading to chisquare tests for quadratic forms in the ::
:: weak inverses of the covariance matrices. Runs are counted ::
:: for sequences of length 10,000. This is done ten times. Then ::
:: repeated. ::
:::
The RUNS test for file test_data.bin
Up and down runs in a sample of 10000

```

---

```

Run test for test_data.bin :
runs up; ks test for 10 p's: .263379
runs down; ks test for 10 p's: .320634
Run test for test_data.bin :
runs up; ks test for 10 p's: .966537
runs down; ks test for 10 p's: .223534

```

\$

```

:::

```



```
:: This is the CRAPS TEST. It plays 200,000 games of craps, finds::
:: the number of wins and the number of throws necessary to end ::
:: each game. The number of wins should be (very close to) a ::
:: normal with mean 200000p and variance 200000p(1-p), with ::
:: p=244/495. Throws necessary to complete the game can vary ::
:: from 1 to infinity, but counts for all>21 are lumped with 21. ::
:: A chi-square test is made on the no.-of-throws cell counts. ::
:: Each 32-bit integer from the test file provides the value for ::
:: the throw of a die, by floating to [0,1), multiplying by 6 ::
:: and taking 1 plus the integer part of the result. ::
:::
Results of craps test for test_data.bin
No. of wins: Observed Expected
                98542    98585.86
                98542= No. of wins, z-score= -.196 pvalue= .42224
Analysis of Throws-per-Game:
Chisq= 20.72 for 20 degrees of freedom, p= .58632
  Throws Observed Expected Chisq Sum
    1 66697 66666.7 .014 .014
    2 37394 37654.3 1.800 1.814
    3 26965 26954.7 .004 1.817
    4 19250 19313.5 .209 2.026
    5 13861 13851.4 .007 2.033
    6 9959 9943.5 .024 2.057
    7 7298 7145.0 3.275 5.332
    8 5212 5139.1 1.035 6.367
    9 3721 3699.9 .121 6.488
   10 2733 2666.3 1.669 8.156
   11 1877 1923.3 1.116 9.272
   12 1353 1388.7 .920 10.192
   13 1057 1003.7 2.829 13.021
   14 757 726.1 1.311 14.332
   15 495 525.8 1.808 16.141
   16 353 381.2 2.079 18.220
   17 274 276.5 .023 18.243
   18 209 200.8 .332 18.575
   19 154 146.0 .440 19.015
   20 113 106.2 .433 19.449
   21 268 287.1 1.273 20.722
SUMMARY FOR test_data.bin
p-value for no. of wins: .422241
p-value for throws/game: .586316

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```



## 9. References

---

- UM10430: LPC18xx User Manual.
- A Statistical Test Suite for Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications.
- The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness.



## 10. Legal information

### 10.1 Definitions

**Draft** — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

### 10.2 Disclaimers

**Limited warranty and liability** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**Right to make changes** — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on a weakness or default in the customer application/use or the application/use of customer's third party customer(s) (hereinafter both referred to as "Application"). It is customer's sole responsibility to check whether the NXP Semiconductors product is suitable and fit for the Application planned. Customer has to do all necessary testing for the Application in order to avoid a default of the Application and the product. NXP Semiconductors does not accept any liability in this respect.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out of the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US\$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

### 10.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.



## 11. Contents

<b>1.</b>	<b>Introduction .....</b>	<b>3</b>	<b>10.3</b>	<b>Trademarks .....</b>	<b>41</b>
<b>2.</b>	<b>True Random Number Generator (TRNG) .....</b>	<b>3</b>	<b>11.</b>	<b>Contents .....</b>	<b>42</b>
<b>3.</b>	<b>Generation of Random numbers using TRNG block.....</b>	<b>4</b>			
<b>4.</b>	<b>NIST SP800-22 test suite.....</b>	<b>6</b>			
4.1	Introduction .....	6			
4.2	NIST SP800-22 test suite description .....	6			
4.2.1	Frequency (Monobit) Test .....	6			
4.2.2	Frequency test within a Block.....	7			
4.2.3	Runs test.....	7			
4.2.4	Test for the Longest run of Ones in a Block .....	7			
4.2.5	Binary Matrix Rank Test.....	7			
4.2.6	Linear Complexity Test .....	7			
4.2.7	Discrete Fourier Transform (Spectral) Test.....	7			
4.2.8	Non-overlapping Template Matching Test .....	7			
4.2.9	Overlapping Template Matching Test .....	7			
4.2.10	Maurer's "Universal Statistical" Test.....	7			
4.2.11	Serial Test .....	7			
4.2.12	Approximate Entropy Test.....	7			
4.2.13	Cumulative Sums (Cusum) Test .....	8			
4.2.14	Random Excursions Test.....	8			
4.2.15	Random Excursions Variant Test.....	8			
<b>5.</b>	<b>Analyzing the NIST SP800-22 Test Suite .....</b>	<b>8</b>			
5.1	Running NIST Test Suite .....	8			
5.1.1	Step 1.....	8			
5.1.2	Step 2.....	9			
5.1.3	Step 3.....	9			
5.1.4	Step 4.....	10			
5.1.5	Step 5.....	10			
5.1.6	Test Report .....	10			
<b>6.</b>	<b>DIEHARD: A Battery of Tests of Randomness</b>	<b>12</b>			
6.1	Running the DIEHARD Test Suite.....	12			
6.1.1	Step 1.....	12			
6.1.2	Step 2.....	13			
6.1.3	Test report.....	14			
<b>7.</b>	<b>Conclusion.....</b>	<b>14</b>			
<b>8.</b>	<b>Appendix A .....</b>	<b>15</b>			
8.1	Test Report for NIST SP800-22 Test Suite .....	15			
8.2	Test Report for DIEHARD Test Suite .....	21			
<b>9.</b>	<b>References .....</b>	<b>40</b>			
<b>10.</b>	<b>Legal information .....</b>	<b>41</b>			
10.1	Definitions .....	41			
10.2	Disclaimers.....	41			

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.