Application Report
SLAA725–January 2017

Random Number Generation Using MSP430FR59xx and MSP430FR69xx Microcontrollers

Akash Patel, Caleb Overbay *MSP430 Applications*

ABSTRACT

Random number generation has a role in a variety of applications, such as cryptography and tamper detection.

In digital systems, it becomes difficult to introduce the concept of true randomness as a machine executes code in the sequence it is programed. This introduces the notion of true random number generators (TRNGs) and pseudorandom number generators (PRNGs), also known as deterministic random bit generators (DRBGs). TRNGs use some source of entropy to provide for randomness in the system, while PRNGs rely on a seed to generate a sequence of numbers that can be realized as deterministic (that is, starting with the same seed will produce the same set of numbers).

This application report reviews an implementation of a Counter Mode Deterministic Random Byte Generator (CTR-DRBG) on MSP430FR59xx and MSP430FR69xx microcontrollers (MCUs). A C code implementation along with an example of its use can be found in FRAM Utilities.

Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

This application report discusses the implementation of a Counter Mode Deterministic Random Byte Generator (CTR-DRBG) on the MSP430FR59xx and MSP430FR69xx microcontrollers. The implementation utilizes the unique 128-bit true random seed found in the Device Descriptor Information (TLV) as a source of entropy into a block cipher algorithm. Subsequently, pseudorandom bytes can be generated. This CTR-DRBG has been developed in accordance with Section 10.2 of NIST SP 800-90A Rev. 1 using the AES-128 block cipher algorithm. In this document, the National Institute of Standards and Technology (NIST) provides various techniques to generate random bytes in a deterministic fashion dependent on a sufficiently random entropy source.

The MSP430FR59xx and MSP430FR69xx microcontrollers include an AES256 Hardware Accelerator (see the *MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide*). This dedicated accelerator can perform encryption (and decryption) of 128-bit data with 128-bit (as well as 192-bit and 256-bit) keys. This implementation can be done on other families of devices; however, using software to do the encryption proves less efficient than running it on dedicated hardware. This family also enables the benefits of FRAM, which are discussed in Section 4.

The correctness of the implementation in FRAM Utilities was verified using the DRBG test vectors provided by NIST that meet the appropriate parameter requirements of the CTR-DRBG discussed in this document. These parameters include AES-128 encryption, using a derivation function, no prediction resistance, no reseeding, no additional input, and no personalization string. These vectors can be found by going to csrc.nist.gov and searching "CAVP Testing: Random Number Generators".

2 Overview of Operation

The following sections describe the operation of the CTR-DRBG implementation provided in the accompanying software package. The CTR-DRGB was standardized by NIST and a more detailed description can be found in Section 10.2 of NIST SP 800-90A Rev. 1.

Figure 1 shows the basic process for generating pseudorandom bytes. Before generating anything, the working state must always be loaded from FRAM. Then, depending on necessity, the device is instantiated. Instantiation is a one-time process that must occur at the creation of the CTR-DRBG. After the device has been instantiated, pseudorandom bytes can be generated. Finally after generation, the working state must be saved to FRAM to reflect updates that took place during the generation process. Loading from and saving to FRAM is a crucial part of this implementation. The full benefits of FRAM in this application are further explored in Section 4.

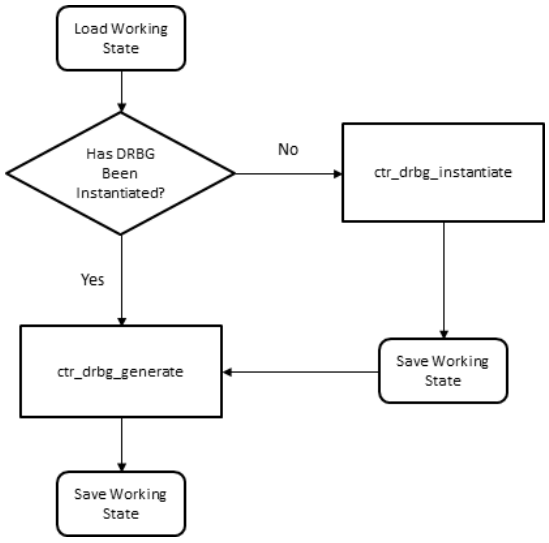

Figure 1. Basic Process of Generating Pseudorandom Bytes

Table 1 lists the essential parameters and their required bit lengths according to the NIST publication mentioned previously. These parameters are discussed and used throughout this document.

Table 1. CTR-DRBG Parameter Requirements

Parameter	Requirement
Input and Output Block Length (*blocklen*)	128 bits
Key Length (*keylen*)	128 bits
Seed Length (*seedlen* = *outlen* + *keylen*)	256 bits
Entropy Input Length	seedlen
Max Number of Bits per Request	2^{11} bits
Maximum Requests Between Reseeds (*reseed_interval*)	2^{48} requests

2.1 CTR-DRBG Working State

The CTR-DRBG is based on a working state that is loaded from and saved to FRAM before and after any pseudorandom bytes are generated. This working state consists of a *key, data (v), and instantiated_flag.* The *key* and *data (v)* are used as inputs to the block encrypt algorithm (AES-128) while the *instantiated_flag* is used to signal whether or not the CTR-DRBG has been instantiated on the device before.

By default, the accompanying software package stores the working state in the FRAM section InfoMemD. The working state holds the "secret values" *key* and *v* that are critical to maintaining the security of the CTR-DRBG. For this reason, if your application requires additional security, TI recommends locking the JTAG after the device has been programmed to keep the working state from being accessed outside the device. See Section 5.2 for more information regarding locking the JTAG.

NOTE: The reseed counter as described in the NIST publication is not realized in this implementation. Section 2.7 includes more information on this and any implications.

2.2 Source of Entropy and Nonce

The MSP430FR59xx and MSP430FR69xx MCUs provide a 128-bit true random seed found in the Device Descriptor Information (TLV) at hex address 0x1A30. This seed is used as the entropy input to instantiate the CTR-DRBG.

The CTR-DRBG also requires a nonce at the time of instantiation. According to the NIST publication, the nonce must be a value that is expected to repeat no more than a 64-bit random string would be expected to repeat. A value fitting this description can be found in the Device Descriptor Information (TLV) and includes the lot and wafer ID, die x position, and die y position. The combination of these three values is unique to every device and provides a sufficient nonce for instantiation.

2.3 Instantiation

The CTR-DRBG requires instantiation to initialize the working state before any pseudorandom bytes can be generated. The instantiation process does several things, including obtaining sufficient entropy input and initializing the working state.

Figure 2 shows the basic process undergone when instantiating the CTR-DRBG. When obtaining the source of entropy, the instantiation function retrieves the true random seed found in the Device Descriptor Information (TLV). However, this seed is only a 128-bit value and, as described in Table 1, a sufficient entropy source must be 256 bits in length. The expansion of the seed into a 256-bit value is realized through the use of a derivation function discussed in Section 2.4.

This entropy source is then used to initialize the working state with the use of the update process described in Section 2.5.

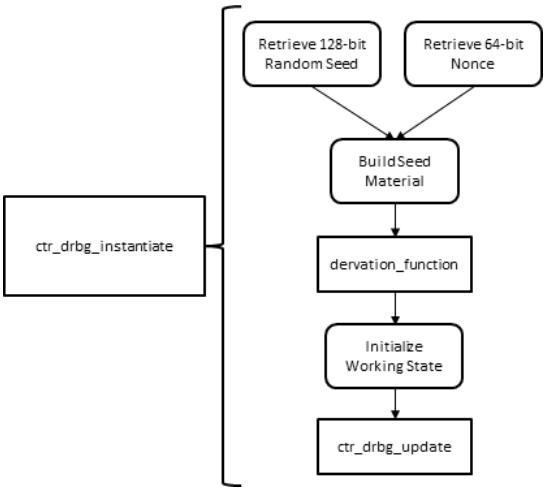

Figure 2. Basic Process of Instantiation

2.4 Derivation Function

The MSP430FR59xx and MSP430FR69xx MCUs can only provide a 128-bit true random source of entropy, but 256 bits are required to ensure the CTR-DRBG meets security requirements. To expand this value to sufficient length, a derivation function requiring a nonce must be used. This derivation function takes advantage of the AES256 accelerator on this family of devices to distribute the 128-bit entropy throughout the wider bit string and provides a suitable means of utilizing the 128-bit true random seed found in the TLV.

2.5 Update Process

At the heart of instantiation and generating pseudorandom bytes is an update function that updates the working state of the CTR-DRBG using *provided_data* as well as AES-128 encryption. This function is responsible for the actions listed in Figure 3.

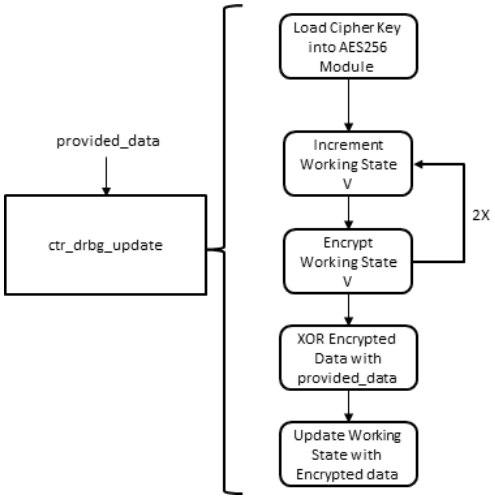

Figure 3. Basic Update Process

The *provided_data* is the source of entropy when update is being used for instantiation. Otherwise, the *provided_data* is zero. According to the NIST standard, the *provided_data* could also be additional input. However, this feature is optional and not supported in the accompanying software package and is why the *provided_data* is typically zero.

2.6 Generating Pseudorandom Bytes

Similar to the update process, generating pseudorandom bytes involves the use of AES-128 encryption to populate an array of random bytes. This function is responsible for the actions shown in Figure 4.

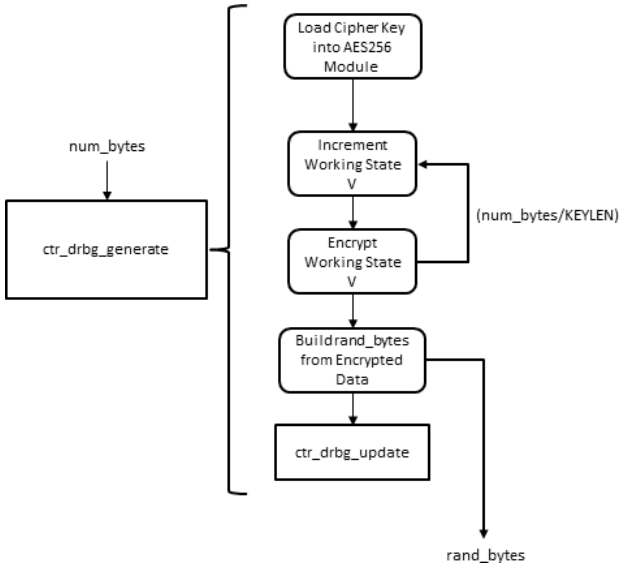

Figure 4. Basic Generation Process

As can be seen, the generation process is very similar to the update process. Generation involves the following steps:

1. Load a cipher key
2. Loop through the steps of incrementing *data (v)* and encryption
3. Build the random bytes
4. Use the update process to update the working state

The *provided_data* passed to the update process is zero due to this implementation not supporting additional input.

2.7 Reseeding

Every time the CTR-DRBG is instantiated or reseeded, a new source of entropy is required. The true random seed found in the MSP430FR59xx and MSP430FR69xx MCUs is programmed during production and cannot be updated or changed. For this reason, the CTR-DRBG implementation in the accompanying software package does not support reseeding.

Even though reseeding is not supported, Table 1 shows that the maximum requests between reseeds is 2^{48}. This number is much larger than even the write endurance of the FRAM at 10^{14} and is sufficient for most if not all applications.

3 Using the Software Package

To generate pseudorandom bytes, call the *rng_generateBytes* function with the number of requested bytes and a pointer to where they should be stored. The function can generate random bytes only in multiples of *keylen*. Figure 1 shows a diagram of the internal workings of the *rng_generateBytes* function and how it generates pseudorandom bytes. Also, Figure 12 in Section 10.2.1 of NIST SP 800-90A Rev. 1 provides a more detailed view of this software flow diagram. See the *FRAM Utilities Users Guide* for more information on how to use the API.

4 Benefits of FRAM

FRAM (Ferroelectric Random Access Memory) provides key benefits in this random number generation application. As it is necessary to save the state of the CTR-DRBG in nonvolatile memory, FRAM is the better choice over flash or EEPROM due to its faster read and write times, lower power consumption, and better data reliability (see FRAM FAQs for more information). These benefits help maintain the working state of the CTR-DRBG in case of power failure or other unexpected device resets.

5 Security of the CTR-DRBG

This implementation of the CTR-DRBG meets the minimum security requirements recommended by the NIST publication. However, there are still ways a potential threat could compromise security by accessing the working state, random seed, and nonce through JTAG. Also, even though this algorithm has been verified as correct and meets the minimum security requirements, not all features described in the NIST publication are implemented. This section describes what features are not present, why those decisions were made, and how to make it more difficult for external threats to access the working state, random seed, and nonce.

NOTE: Throughout the process of creating pseudorandom numbers, values critical to security are placed on the stack. Where applicable in the accompanying software package, these values are cleared before returning from the function in which they are created.

5.1 Personalization String, Reseeding, and Prediction Resistance

This implementation does not support personalization strings, reseeding, or prediction resistance as described in the NIST publication. See Section 2.7 for an explanation of why this reseeding is disabled. The personalization string is an optional feature and is not implemented to reduce the burden on the consuming application. Finally, prediction resistance is not implemented because of the need to constantly reseed when this feature is available.

5.2 Access to the Working State and JTAG Lock

The working state of the CTR-DRBG holds the "secret values" *key* and *data (v)* that are crucial to the security of the numbers generated. Access to these would allow prediction and backtracking of the pseudorandom numbers generated. Also, access to the 128-bit random seed and nonce located in the Device Descriptor Table would allow similar prediction and backtracking.

The MSP430FR59xx and MSP430FR69xx MCUs can lock the JTAG either with or without a password. This is a way of keeping data on the devices from being read through the JTAG interface. While locking the JTAG is not a perfect solution to protecting the working state, nonce, and random seed from unauthorized access, the lock makes access much more difficult.

If you are concerned about the security of your application that is using this CTR-DRBG, TI highly recommends taking advantage of the JTAG locking feature. More information regarding this feature can be found in *MSP430 Programming With the JTAG Interface* and *MSP Code Protection Features*.

6 Degree of Randomness

There are several common ways to extensively test the randomness of data generated by a pseudorandom number generator. The Diehard suite and the NIST suite are both widely used batteries of statistical tests that verify the quality of a pseudorandom number generator. Both of these test suites were run on the CTR-DRBG described in this document and the results suggest the data generated is close to true randomness. To run these tests, 23.4 million random bytes were generated on an MSP430FR5969 MCU. These bytes were then used as input to the suites. Further information on the individual suites as well as raw data and test results can be found in the following sections and appendices.

6.1 NIST Test Suite

The CTR-DRBG in the accompanying software package was subjected to the NIST test suite described in NIST SP 800-22 Revision 1A. This suite consists of fifteen statistical tests each returning a p-value. The data provided to the tests was broken into eight binary sequences for the random excursion tests and ten binary sequences for all others. The p-values were analyzed by the test suite and a clear pass or fail result was given.

For a test to be considered successful, at least 7 out of 8 binary sequences must pass the random excursion tests, and eight out of ten for everything else. All tests passed these criteria, strongly suggesting that the pseudorandom numbers generated by the CTR-DRBG are very close to being truly random. Appendix A lists the raw data organized by the test that produced the result.

6.2 Diehard Test Suite

The CTR-DRBG implemented in the accompanying software package was also subjected to the Diehard suite of tests. This suite consists of 12 statistical tests, and each returns a varying number of p-values. For the data to be truly random, these p-values should be evenly distributed on [0, 1). This means that ten percent of the p-values output from the test suite should be found from 0.0 to 0.1, another ten percent from 0.1 to 0.2, and so on.

Table 2 is an interpretation of the results of the Diehard test suite. As can be seen, in each 0.1 increment, ten percent of the p-values are expected to be observed. These observed values followed this trend very closely with the lowest percentage seen being 7.44% and the highest being 13.02%. Unfortunately, the tests do not define a margin that the numbers should fall within to classify them as passing and is why the NIST test suite was also run on this data.

Table 2. CTR-DRBG Diehard P-Value Distribution Results

P-Value Range	Observed Percent	'Expected' Percent
0.0–0.1	8.84%	10%
0.1–0.2	13.02%	10%
0.2–0.3	9.77%	10%
0.3–0.4	9.77%	10%
0.4–0.5	11.16%	10%
0.5–0.6	8.84%	10%
0.6–0.7	9.30%	10%
0.7–0.8	12.09%	10%
0.8–0.9	9.77%	10%
0.9–1.0	7.44%	10%

In a more compelling fashion, Figure 5 reiterates what can be seen in Table 2. The red line represents the ideal distribution of p-values, and the blue line represents every observed p-value. Although the observed results are not perfect, they show a strong indication that the implementation discussed in this document is producing pseudorandom numbers that are very close to ideal. Appendix B lists the raw data produced by the Diehard suite, including p-values and the tests they were generated from.

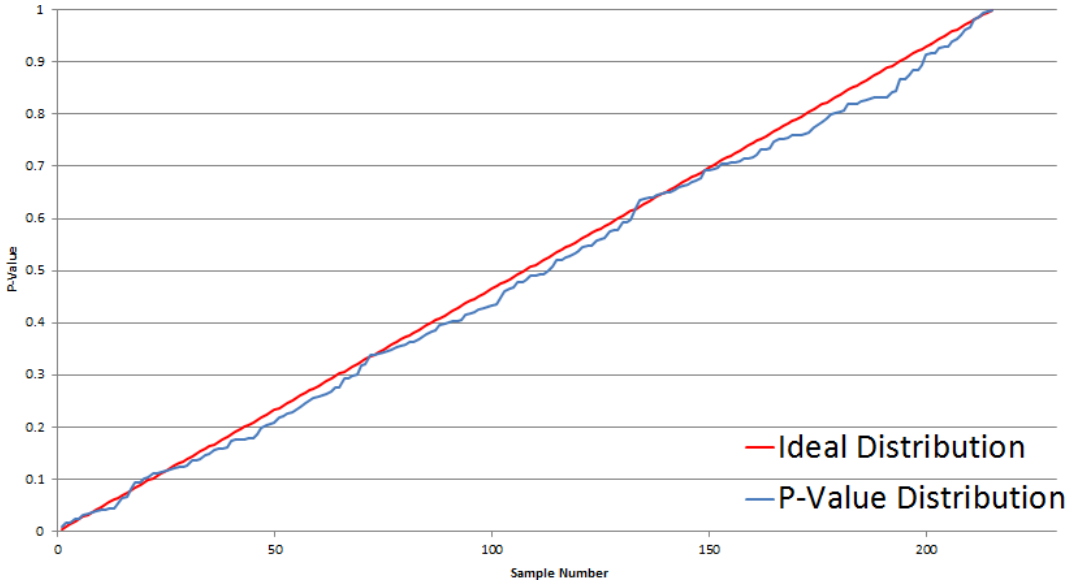

Figure 5. CTR-DRBG Diehard P-Value Distribution Results

7 References

- *MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide*
- *FRAM FAQs*
- *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* (NIST Special Publication 800-90A, Revision 1)
- *MSP Code Protection Features*
- *MSP430 Programming With the JTAG Interface*
- *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications* (NIST Special Publication 800-22, Revision 1A)

Raw NIST Test Data

A.1 NIST Test Suite Data

```
---------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
---------------------------------------------------------------------------
   generator is <rand_nums.32%gt;
---------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
---------------------------------------------------------------------------
  2   1   2   0   1   1   1   2   0   0  0.739918     10/10      Frequency
  0   2   2   1   0   0   0   1   2   2  0.534146     10/10      BlockFrequency
  2   1   1   0   1   0   1   0   3   1  0.534146     10/10      CumulativeSums
  2   1   1   1   1   0   1   1   0   2  0.911413     10/10      CumulativeSums
  0   2   1   1   3   0   1   1   0   1  0.534146     10/10      Runs
  1   1   0   1   1   4   0   1   0   1  0.213309     10/10      LongestRun
  3   0   2   1   0   0   0   0   0   4  0.017912     10/10      Rank
  1   2   0   0   1   2   2   1   1   0  0.739918      9/10      FFT
  2   2   0   0   3   1   0   0   1   1  0.350485     10/10      NonOverlappingTemplate
  0   2   2   1   0   2   2   1   0   0  0.534146     10/10      NonOverlappingTemplate
  0   2   2   2   1   0   1   2   0   0  0.534146     10/10      NonOverlappingTemplate
  1   1   2   1   3   0   1   1   0   0  0.534146     10/10      NonOverlappingTemplate
  1   0   0   2   3   3   0   1   0   0  0.122325     10/10      NonOverlappingTemplate
  2   1   0   3   1   0   0   0   2   1  0.350485     10/10      NonOverlappingTemplate
  0   0   3   0   0   0   3   1   1   2  0.122325     10/10      NonOverlappingTemplate
  2   2   0   0   2   0   1   1   2   0  0.534146     10/10      NonOverlappingTemplate
  1   1   2   1   2   0   2   0   0   1  0.739918     10/10      NonOverlappingTemplate
  2   1   1   0   1   2   0   2   1   0  0.739918      9/10      NonOverlappingTemplate
  0   2   0   1   0   1   1   1   4   0  0.122325     10/10      NonOverlappingTemplate
  3   0   0   0   0   2   1   2   2   0  0.213309     10/10      NonOverlappingTemplate
  1   1   0   0   1   0   3   4   0   0  0.035174      9/10      NonOverlappingTemplate
  0   0   1   4   0   2   1   1   0   1  0.122325     10/10      NonOverlappingTemplate
  2   1   1   1   0   2   0   1   1   1  0.911413     10/10      NonOverlappingTemplate
  2   1   2   1   1   0   0   0   1   2  0.739918     10/10      NonOverlappingTemplate
  1   0   2   1   0   0   1   1   2   2  0.739918     10/10      NonOverlappingTemplate
  2   1   0   0   2   3   0   1   1   0  0.350485     10/10      NonOverlappingTemplate
  1   1   0   1   0   1   2   2   0   2  0.739918     10/10      NonOverlappingTemplate
  0   2   0   0   1   1   2   1   0   3  0.350485     10/10      NonOverlappingTemplate
  0   0   0   1   0   1   2   0   1   5  0.008879     10/10      NonOverlappingTemplate
  3   0   1   0   0   1   2   1   0   2  0.350485     10/10      NonOverlappingTemplate
  1   2   2   1   1   0   2   0   0   1  0.739918     10/10      NonOverlappingTemplate
  3   2   1   0   2   0   1   1   0   0  0.350485      9/10      NonOverlappingTemplate
  1   0   0   4   0   0   2   0   2   1  0.066882     10/10      NonOverlappingTemplate
  0   0   1   1   1   1   1   1   3   1  0.739918     10/10      NonOverlappingTemplate
  1   0   4   1   1   0   0   1   0   2  0.122325     10/10      NonOverlappingTemplate
  0   1   0   0   0   2   1   3   2   1  0.350485     10/10      NonOverlappingTemplate
  1   1   0   1   4   2   0   1   0   0  0.122325      9/10      NonOverlappingTemplate
  3   0   1   1   1   0   0   0   2   2  0.350485      9/10      NonOverlappingTemplate
  0   0   3   0   3   1   0   2   0   1  0.122325     10/10      NonOverlappingTemplate
  1   2   1   1   2   1   0   0   1   1  0.911413     10/10      NonOverlappingTemplate
  2   0   2   1   1   1   1   1   1   0  0.911413     10/10      NonOverlappingTemplate
  2   1   1   0   2   0   1   0   1   2  0.739918     10/10      NonOverlappingTemplate
  2   1   1   0   0   2   1   1   2   0  0.739918     10/10      NonOverlappingTemplate
  2   0   2   2   1   0   2   0   1   0  0.534146     10/10      NonOverlappingTemplate
  3   0   0   3   1   1   0   0   1   1  0.213309      9/10      NonOverlappingTemplate
```

1	0	0	2	2	1	1	0	2	1	0.739918	10/10	NonOverlappingTemplate
0	0	0	1	1	1	0	4	2	1	0.122325	10/10	NonOverlappingTemplate
0	1	0	0	1	1	1	5	1	0	0.017912	10/10	NonOverlappingTemplate
2	0	2	0	3	0	1	0	2	0	0.213309	10/10	NonOverlappingTemplate
0	1	1	1	1	3	0	0	2	1	0.534146	10/10	NonOverlappingTemplate
0	0	0	1	1	1	1	2	3	1	0.534146	10/10	NonOverlappingTemplate
2	2	2	2	0	0	1	0	1	0	0.534146	10/10	NonOverlappingTemplate
0	0	1	4	2	0	2	0	1	0	0.066882	10/10	NonOverlappingTemplate
0	2	0	1	1	0	4	0	0	2	0.066882	10/10	NonOverlappingTemplate
2	1	2	0	1	0	1	0	1	2	0.739918	10/10	NonOverlappingTemplate
0	2	1	0	2	2	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
1	0	1	0	1	0	1	5	1	0	0.017912	10/10	NonOverlappingTemplate
2	1	1	0	1	1	0	1	1	2	0.911413	9/10	NonOverlappingTemplate
1	1	0	1	2	2	0	1	0	2	0.739918	10/10	NonOverlappingTemplate
0	1	2	0	0	0	3	0	2	2	0.213309	10/10	NonOverlappingTemplate
0	0	0	0	0	2	6	0	2	0	0.000089 *	10/10	NonOverlappingTemplate
0	0	2	1	1	2	1	1	1	1	0.911413	10/10	NonOverlappingTemplate
2	1	3	2	1	1	0	0	0	0	0.350485	10/10	NonOverlappingTemplate
1	3	1	1	1	2	0	1	0	0	0.534146	10/10	NonOverlappingTemplate
0	1	0	5	0	0	2	1	1	0	0.008879	10/10	NonOverlappingTemplate
4	0	0	1	0	0	2	2	0	1	0.066882	10/10	NonOverlappingTemplate
0	2	0	0	2	1	2	0	1	2	0.534146	10/10	NonOverlappingTemplate
1	1	0	2	2	1	0	2	1	0	0.739918	10/10	NonOverlappingTemplate
0	2	2	1	0	2	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
2	0	1	0	0	1	3	1	0	2	0.350485	9/10	NonOverlappingTemplate
2	3	0	1	1	1	1	0	1	0	0.534146	10/10	NonOverlappingTemplate
0	1	1	2	0	1	1	0	1	3	0.534146	10/10	NonOverlappingTemplate
0	0	1	3	1	0	0	3	1	1	0.213309	10/10	NonOverlappingTemplate
2	1	1	0	0	0	0	1	2	3	0.350485	10/10	NonOverlappingTemplate
0	1	2	1	0	2	1	1	1	1	0.911413	10/10	NonOverlappingTemplate
1	1	0	1	3	0	0	0	2	2	0.350485	10/10	NonOverlappingTemplate
1	0	3	1	0	2	0	1	1	1	0.534146	9/10	NonOverlappingTemplate
1	1	2	1	1	0	2	0	1	1	0.911413	10/10	NonOverlappingTemplate
0	2	1	1	2	1	0	0	2	1	0.739918	10/10	NonOverlappingTemplate
2	1	2	2	0	1	0	2	0	0	0.534146	9/10	NonOverlappingTemplate
0	2	0	1	1	0	2	1	3	0	0.350485	10/10	NonOverlappingTemplate
0	3	0	1	2	2	0	1	1	0	0.350485	10/10	NonOverlappingTemplate
2	2	0	0	3	1	0	0	1	1	0.350485	10/10	NonOverlappingTemplate
2	0	2	1	1	0	0	1	2	1	0.739918	10/10	NonOverlappingTemplate
0	0	2	0	2	2	2	2	0	0	0.350485	10/10	NonOverlappingTemplate
0	1	2	2	2	0	1	2	0	0	0.534146	10/10	NonOverlappingTemplate
2	2	0	0	0	0	1	3	1	1	0.350485	10/10	NonOverlappingTemplate
1	1	1	3	1	1	1	1	0	0	0.739918	10/10	NonOverlappingTemplate
2	0	0	3	2	0	1	1	1	0	0.350485	10/10	NonOverlappingTemplate
0	2	1	1	2	0	0	3	1	0	0.350485	10/10	NonOverlappingTemplate
0	1	1	2	0	0	2	1	2	1	0.739918	10/10	NonOverlappingTemplate
0	1	0	2	2	0	1	1	1	2	0.739918	10/10	NonOverlappingTemplate
0	2	1	1	0	1	0	1	3	1	0.534146	10/10	NonOverlappingTemplate
0	1	1	3	1	1	0	1	2	0	0.534146	10/10	NonOverlappingTemplate
1	3	0	2	0	1	1	2	0	0	0.350485	10/10	NonOverlappingTemplate
5	0	1	0	0	0	0	2	0	2	0.004301	9/10	NonOverlappingTemplate
2	1	1	1	0	2	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
0	2	0	0	0	1	1	2	3	1	0.350485	10/10	NonOverlappingTemplate
0	1	1	0	2	1	3	1	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	0	1	1	1	1	0	3	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	0	1	2	0	1	1	2	0.911413	10/10	NonOverlappingTemplate
1	0	2	0	2	0	3	0	2	0	0.213309	10/10	NonOverlappingTemplate
0	1	2	0	3	1	1	0	2	0	0.350485	10/10	NonOverlappingTemplate
2	1	2	2	1	0	1	1	0	0	0.739918	10/10	NonOverlappingTemplate
0	1	1	2	2	2	0	2	0	0	0.534146	10/10	NonOverlappingTemplate
1	1	1	1	1	0	0	5	0	0	0.017912	10/10	NonOverlappingTemplate
0	1	1	1	1	2	0	1	2	1	0.911413	10/10	NonOverlappingTemplate
0	1	0	0	2	2	2	2	0	1	0.534146	10/10	NonOverlappingTemplate
1	0	0	0	1	1	5	2	0	0	0.008879	10/10	NonOverlappingTemplate
4	1	0	0	3	0	1	0	0	1	0.035174	9/10	NonOverlappingTemplate
0	1	0	3	0	0	2	3	0	1	0.122325	10/10	NonOverlappingTemplate

0	1	1	1	3	2	1	0	0	1	0.534146	10/10	NonOverlappingTemplate
0	1	2	0	1	1	2	1	0	2	0.739918	10/10	NonOverlappingTemplate
0	0	1	2	2	2	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
3	0	0	3	2	0	0	1	0	1	0.122325	10/10	NonOverlappingTemplate
2	0	1	0	0	1	1	0	3	2	0.350485	10/10	NonOverlappingTemplate
1	0	4	0	0	1	2	2	0	0	0.066882	10/10	NonOverlappingTemplate
1	2	0	1	2	1	2	0	1	0	0.739918	10/10	NonOverlappingTemplate
0	0	1	2	1	0	2	1	1	2	0.739918	10/10	NonOverlappingTemplate
0	1	1	0	2	2	1	1	1	1	0.911413	10/10	NonOverlappingTemplate
1	2	1	1	2	0	0	2	0	1	0.739918	10/10	NonOverlappingTemplate
2	0	0	3	0	1	1	0	2	1	0.350485	10/10	NonOverlappingTemplate
1	2	0	0	1	2	0	1	2	1	0.739918	10/10	NonOverlappingTemplate
0	3	0	0	2	3	1	0	0	1	0.122325	10/10	NonOverlappingTemplate
0	0	2	1	1	0	1	1	2	2	0.739918	10/10	NonOverlappingTemplate
0	0	0	1	2	1	2	2	1	1	0.739918	10/10	NonOverlappingTemplate
2	2	0	2	1	2	1	0	0	0	0.534146	9/10	NonOverlappingTemplate
1	4	0	2	0	0	0	1	1	1	0.122325	10/10	NonOverlappingTemplate
0	2	0	1	1	1	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
1	0	2	1	1	1	3	0	0	1	0.534146	10/10	NonOverlappingTemplate
2	0	0	1	1	1	2	1	1	1	0.911413	10/10	NonOverlappingTemplate
0	1	2	1	1	1	0	1	2	1	0.911413	10/10	NonOverlappingTemplate
1	2	0	1	0	1	2	1	2	0	0.739918	9/10	NonOverlappingTemplate
0	1	1	0	2	1	1	0	3	1	0.534146	10/10	NonOverlappingTemplate
2	4	0	0	0	0	0	3	1	0	0.017912	9/10	NonOverlappingTemplate
1	0	0	0	2	1	3	1	0	2	0.350485	10/10	NonOverlappingTemplate
1	2	1	2	0	1	0	2	0	1	0.739918	10/10	NonOverlappingTemplate
3	1	1	2	0	0	2	1	0	0	0.350485	10/10	NonOverlappingTemplate
1	1	1	0	2	1	1	1	2	0	0.911413	10/10	NonOverlappingTemplate
1	2	0	1	2	0	3	0	0	1	0.350485	10/10	NonOverlappingTemplate
2	2	1	0	0	1	1	2	1	0	0.739918	9/10	NonOverlappingTemplate
1	0	1	3	1	0	1	0	2	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	2	1	0	3	2	0	1	0.350485	10/10	NonOverlappingTemplate
1	4	1	0	0	0	1	1	1	1	0.213309	10/10	NonOverlappingTemplate
2	1	0	0	3	2	1	0	1	0	0.350485	10/10	NonOverlappingTemplate
0	0	1	2	1	2	0	1	2	1	0.739918	10/10	NonOverlappingTemplate
2	2	0	0	0	1	2	1	1	1	0.739918	10/10	NonOverlappingTemplate
2	2	0	2	0	0	2	1	0	1	0.534146	9/10	NonOverlappingTemplate
2	1	0	0	1	1	2	1	1	1	0.911413	10/10	NonOverlappingTemplate
3	0	1	1	1	2	0	1	0	1	0.534146	10/10	NonOverlappingTemplate
0	0	1	3	1	1	0	1	1	2	0.534146	10/10	NonOverlappingTemplate
3	0	1	2	2	0	1	1	0	0	0.350485	9/10	NonOverlappingTemplate
2	1	0	0	0	2	2	1	0	2	0.534146	10/10	NonOverlappingTemplate
0	0	1	1	1	4	2	1	0	0	0.122325	10/10	NonOverlappingTemplate
0	0	3	1	1	0	1	2	1	1	0.534146	10/10	NonOverlappingTemplate
0	3	0	1	2	2	0	1	1	0	0.350485	10/10	NonOverlappingTemplate
0	3	1	1	0	1	1	1	1	1	0.739918	10/10	OverlappingTemplate
0	2	2	2	0	2	2	0	0	0	0.350485	10/10	Universal
1	0	3	0	2	1	2	0	1	0	0.350485	10/10	ApproximateEntropy
1	1	0	0	1	1	2	1	1	0	----	7/8	RandomExcursions
1	0	0	1	1	0	1	0	2	2	----	7/8	RandomExcursions
0	1	0	1	2	1	0	0	1	2	----	8/8	RandomExcursions
0	1	0	0	0	1	2	2	1	1	----	8/8	RandomExcursions
1	0	1	1	0	1	1	0	2	1	----	8/8	RandomExcursions
0	1	0	0	3	1	0	2	1	0	----	8/8	RandomExcursions
1	1	2	0	0	2	1	1	0	0	----	8/8	RandomExcursions
1	1	2	2	0	0	0	1	1	0	----	8/8	RandomExcursions
2	3	0	0	0	0	0	2	1	0	----	7/8	RandomExcursionsVariant
3	1	1	0	0	0	0	0	2	1	----	8/8	RandomExcursionsVariant
2	1	1	1	0	0	0	1	1	1	----	8/8	RandomExcursionsVariant
1	0	1	3	0	0	1	1	1	0	----	8/8	RandomExcursionsVariant
1	0	2	1	0	1	2	0	0	1	----	8/8	RandomExcursionsVariant
1	0	0	2	3	0	0	0	0	2	----	8/8	RandomExcursionsVariant
1	0	0	2	1	0	1	3	0	0	----	8/8	RandomExcursionsVariant
0	1	1	0	1	0	1	1	0	3	----	8/8	RandomExcursionsVariant
0	0	2	1	0	2	0	1	1	1	----	8/8	RandomExcursionsVariant
0	1	1	3	2	0	0	0	1	0	----	8/8	RandomExcursionsVariant

```
  0   1   1   1   3   1   0   1   0   0     ----       8/8       RandomExcursionsVariant
  1   1   0   0   2   0   2   0   1   1     ----       8/8       RandomExcursionsVariant
  1   0   1   1   1   0   1   0   1   2     ----       8/8       RandomExcursionsVariant
  1   0   1   1   0   1   2   0   1   1     ----       7/8       RandomExcursionsVariant
  1   0   1   1   0   0   1   3   1   0     ----       8/8       RandomExcursionsVariant
  1   0   0   1   1   0   1   1   1   2     ----       8/8       RandomExcursionsVariant
  1   0   0   1   0   2   0   0   0   4     ----       8/8       RandomExcursionsVariant
  0   1   1   0   0   1   1   2   1   1     ----       8/8       RandomExcursionsVariant
  2   2   0   2   0   2   0   1   0   1  0.534146     10/10      Serial
  3   0   0   1   0   1   1   1   3   0  0.213309     10/10      Serial
  0   2   0   2   2   1   0   0   1   2  0.534146     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 = 7 for a sample size = 8 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Raw Diehard Test Data

B.1 NIST Test Suite Data

Test	P-Values					
Birthday Spacing	0.57692	0.619324	0.024827	0.431826	0.038608	
	0.575724	0.832611	0.986409	0.15656		
Overlapping 5-Permutation	0.035669	0.147976				
Binary Rank Test for 31x31 Matrices	0.395666					
Binary Rank Test for 32x32 Matrices	0.867359					
Binary Rank Test for 6x8 Matrices	0.76077	0.71749	0.15782	0.17364	0.65062	
	0.66416	0.36365	0.26746	0.83291	0.64876	
	0.69357	0.01542	0.20571	0.86797	0.49402	
	0.2764	0.73343	0.41972	0.91576	0.42456	
	0.20789	0.36411	0.17734	0.25563	0.51928	
Bitstream	0.80805	0.12357	0.56282	0.46154	0.81932	
	0.73491	0.99792	0.38662	0.29847	0.80549	
	0.54712	0.67202	0.20255	0.75516	0.63408	
	0.53228	0.30009	0.46804	0.4028	0.01558	
Overlapping Pairs Sparse Occupancy	0.8265	0.7219	0.0086	0.5612	0.1392	0.5936
	0.9399	0.1245	0.4995	0.4899	0.2216	0.2384
	0.1155	0.0547	0.2341	0.1026	0.952	0.1102
	0.7334	0.8444	0.0948	0.6396	0.9936	
Overlapping Quadruples Sparse Occupancy	0.7475	0.9292	0.8206	0.3555	0.7603	0.4941
	0.0338	0.7799	0.9676	0.2922	0.3771	0.5973
	0.3392	0.0811	0.8847	0.7849	0.4163	0.7635
	0.694	0.707	0.4644	0.7994	0.322	
	0.1116	0.024	;0.1222	0.1358	0.4269	
DNA	0.917	0.9806	0.1266	0.347	0.1992	0.9152
	0.8294	0.7657	0.5384	0.5571	0.0419	
	0.677	0.0429	0.5466	0.1784	0.3416	
	0.2283	0.5929	0.662	0.1606	0.1761	
	0.7755	0.8309	0.707	0.7602	0.4178	
	0.2579	0.1777	0.6927	0.1754	0.8309	
Count the 1s	0.639839	0.637017	0.136661	0.353537	0.524947	0.490358
	0.645311	0.225209	0.894011	0.263088	0.404978	0.359262
	0.527177	0.999997	0.400865	0.374214	0.476745	
	0.36752	0.398881	0.337615	0.507594	0.448888	
	0.042498	0.403684	0.714383	0.251934	0.649043	
Parking Lot	0.928018	0.065925	0.276387	0.753306	0.819442	
	0.753306	0.659449	0.842447	0.218799	0.261324	

Test	P-Values				
Minimum Distance Test	0.304473	0.358987	0.296069	0.008007	0.931874
	0.838527	0.461866	0.937948	0.47756	0.851651
	0.12529	0.04575	0.178803	0.020863	0.894159
	0.648139	0.109445	0.911124	0.539006	0.421783
3D Spheres	0.57879	0.70432	0.11952	0.47698	0.15765
	0.92923	0.06343	0.1865	0.09282	0.51931
	0.71436	0.10352	0.70533	0.34363	0.03167
	0.31895	0.54511	0.88435	0.38395	0.7099
Sqeeze	0.433047				
Overlapping Sums	0.669995	0.961213	0.873515	0.347101	0.69825
	0.434938	0.791825	0.823428	0.482064	0.293836
Runs	0.943872	0.655636	0.113751	0.246329	
Craps	0.145735	0.043456			

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265