IBB Ingenieur-Büro Bergmann, Sonnenweg 3, 15537 Grünheide, info@ibbergmann.org

BENUTZERHANDBUCH

des Physikalischen Zufallsgenerators PRG610

Version 1.0

Autor: **Frank Bergmann**

Letzte Änderung: **09.04.2015 11:48**

1 Inhaltsverzeichnis

2 Copyright

IBB Ingenieurbüro Bergmann

Sonnenweg 3

D-15537 Grünheide

Ausgabe vom 09.04.2015

Haftung

Bei der Erarbeitung dieser Dokumentation wurde größter Wert auf die Vollständigkeit und Richtigkeit des Inhalts gelegt. Es kann dennoch keine Garantie für die Vollständigkeit und Richtigkeit übernommen werden.

Für Hinweise zu dieser Dokumentation sind wir dankbar.

Hotline

Die Hotline des Herstellers erreichen Sie unter +49(0)172 308 6554.

Warenzeichen

MS Windows ist eingetragenes Warenzeichen der Microsoft Corp.

3 Bedeutung von Zufallszahlen

Gute Zufallszahlen sind das Fundament vieler kryptographischer Verfahren und Protokolle. Es ist wichtig, dass die verwendeten Zufallszahlen nicht vorhersagbar sind. Solche Zufallszahlen zu erzeugen, fällt Computern naturgemäß schwer. Zahlreiche Meldungen kritisieren Lücken, Schwächen und Manipulationen bei der Erzeugung von Zufallszahlen für kryptografische Verfahren.

Bei allen Meldungen geht es nicht um spezielle, bedeutungslose Applikationen, sondern um millionenfach installierte Standardprogramme in professionellen Anwendungen. Vor allem dort, wo kontinuierlich viele Zufallszahlen benötigt werden (Netzwerke, Kommunikationssysteme), sind statistische Angriffe auf schwache Zufallsgeneratoren am erfolgreichsten.

Nach dem Kerckhoff-Prinzip (die Sicherheit soll nur auf der Geheimhaltung des Schlüssels beruhen, nicht auf der Geheimhaltung des kryptographischen Algorithmus) benötigt jede Art von Verschlüsselung eine geheime Komponente, die unter keinen Umständen vorhersagbar oder rekonstruierbar sein darf: der aus Zufallszahlen gebildete Schlüssel. Diese Zufallszahlen werden in den bekannten IT-Sicherheitsapplikationen aus Pseudozufallszahlen gebildet. Quelle der Generierung von Pseudozufall ist ein so genannter Seed (ein Startwert, bestehend aus Passwort, Timer-Register, Tastaturanschlägen, Mausbewegungen usw.), mit dem ein mathematisch-kryptografischer Algorithmus eine statistisch gut verteilte Zufallsfolge erzeugt. Aber die gesamte Sicherheit der per Pseudozufall erzeugten geheimen Schlüssel hängt *ausschließlich* von dieser Anfangsinitialisierung ab. Die Anfangsinitialisierung ist bei richtiger Wahl der Quelle der einzige wirklich zufällige Parameter, alles Weitere ist *deterministisch* und somit berechenbar. Eine schwache Anfangsinitialisierung (trivialer Seed) ist im statistischen Ergebnis nicht erkennbar, aber ein effizienter Angriffspunkt der Kryptoanalysis.

Auch professionelle Entwickler nutzen als Seed für Pseudozufall oftmals das Timerregister in der Annahme: wer will denn schon wissen, in welcher Sekunde das Register ausgelesen wurde. Für einen Angreifer kein Problem, denn ein Jahr hat ca. 32 Millionen Sekunden. Und um diese mit der totalen Probiermethode (brute force) durchzutesten, benötigt man nur eine durchschnittliche Rechenleistung. Wird der gleiche Seed mehrfach verwendet, so entstehen schlüsselgleiche Geheimtexte. Ein sicherer Erfolg für die Kryptoanalysis.

Der Kryptoanalysis stehen heute wesentlich leistungsfähigere Werkzeuge zur Verfügung, so dass immer häufiger Meldungen zu kompromittierten schwachen Zufallsgeneratoren veröffentlicht werden. Dagegen haben die in IT-Sicherheitslösungen implementierten Pseudozufallsgeneratoren einen Stand erreicht, der eine neue Qualität der Zufallserzeugung erfordert.

Pseudozufall basierende Zufallsgeneratoren sammeln in diversen Entropiequellen in der Hoffnung, davon ausreichende Mengen zu sammeln. Zwar werden in diversen Chip-Sätzen (VIA, Transmeta, Renesas) und Security-Chipkarten derartige Zufallsgeneratoren angeboten, aber über Entropie und Qualität der Zufallsrohdaten werden keine oder nur unzureichende Angaben gemacht. Aus deren Chips konnten Sicherheitsforscher um Bernstein über 80 eindeutige RSA-Schlüssel auslesen, die gemeinsame Primfaktoren haben. Grund dafür war ein fehlerhafter Random Number Generator im AE45C1-Chip von Renesas, der nicht genügend Entropie erzeugt. Die veröffentlichten Daten zur Entropie des VIA C3 PadLock (7,64 Bit/Byte) zeigen für hohe Sicherheitsansprüche nicht ausreichende Werte. Zur Bit-Unabhängigkeit werden keine Informationen aufgeführt. Aber: nichts sagt mehr über die Eigenschaften eines physikalischen Zufallsgenerators aus, wie seine Rohdaten. Jede weitere Verarbeitung der Rohdaten verschleiert nur die wirklich statistischen Basiseigenschaften und kann vor allem die Entropie nicht weiter erhöhen.

4 PRG610

Bei dem PRG610 handelt es sich um einen physikalischen Zufallsgenerator mit einer UART-Schnittstelle mit 3,3V-Pegel. Der PRG610 unterstützt die professionelle Generierung von kryptografisch sicheren Zufallszahlen der Klasse PTG.3. Vorzugsweise ist der PRG610 für den Einsatz auf den Raspberry-Pi-Boards vorgesehen, kann aber auch auf jeder anderen Pin-kompatiblen Plattform eingesetzt werden.

Mittels simpler Kommandos können Zufallsrohdaten zur Ermittlung der Entropie und kryptografisch sichere Zufallszahlen der Klasse PTG.3 erzeugt werden. Verschiedene Sicherheitsfunktionen garantieren, dass nur gleichverteile und unabhängige Zufallszahlen ausgegeben werden. Wird ein Fehler der Rauschquelle oder statistisch auffällige interne Tests erkannt, wird die Zufallsausgabe blockiert und ein „Intensiver Selbsttest" solange aktiviert, bis alle Parameter sich wieder im vorgegebenen Bereich befinden.

Abbildung: PRG610 auf einem Raspberry-Pi-Board (rot umrandet)

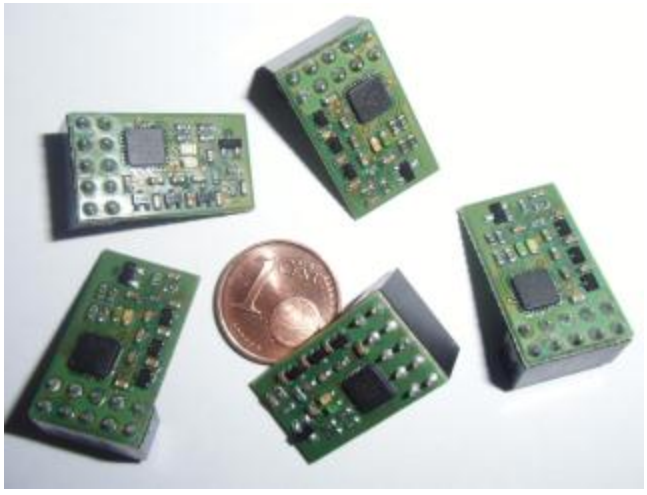


Abbildung: Exemplare des PRG610

5 Technische Daten

Abmessungen:	22*13*12,0 mm (mit Pfostensteckverbinder)
Versorgungsspannung:	3,3V (+/- 10%)
Stromaufnahme:	max. 7mA
Temperaturbereich:	funktionell und statistisch stabil von 0°C..+70°C
Schnittstellen:	UART-Interface, 115,2 Kbit/s
Qualitätssicherung:	Tot-Test zur Überwachung der Rauschquellen Online-Test zur statistischen Überwachung des Zufallssignals Abschaltung der Zufallsausgabe bei Ausfall der Rauschquelle oder unzureichender statistischer Qualität des digitalisierten Rauschsignals
Entropie:	>7,997 Bits/Byte (ermittelt aus Zufalls-Rohdaten nach Shannon)
0/1-Verhältnis:	garantiert im Bereich 0,499..0,501 (> 100 KByte)
Datengenerierung:	> 40 Kbit/s

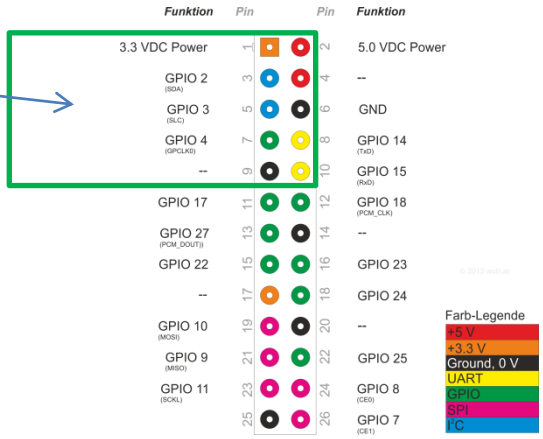
6 Verwendete GPIO des Raspberry-Pi

Der PRG610 nutzt folgende Pins des Raspberry-Boards:

- Pin 1: 3,3V
- Pin 6: GND
- Pin 8: TXD
- Pin 10: RXD

PRG610-Modul

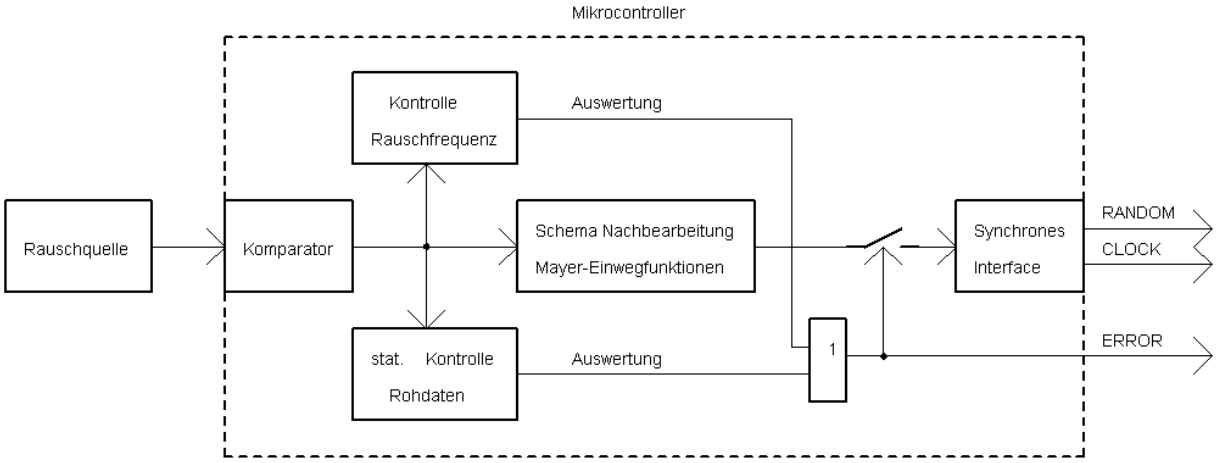
(Draufsicht)



7 Stochastische Modell

Ein stochastisches Modell unterstützt Gutachten über die Qualität und Zuverlässigkeit eines Zufallsgenerators. Es beschreibt die Entropiequelle, die Verarbeitung des digitalisierten Rauschsignals, die kryptografische Nachbereitung und die Sicherheitsfunktionen zur Überwachung der Signalverarbeitung.

Das stochastische Modell des PRG610 wird in folgendem Schema verdeutlicht:



8 Prinzip der Rauscherzeugung des PRG610

Rauschen ist ein physikalisches Phänomen und stellt eine Störgröße mit breitem unspezifischem Frequenzspektrum dar. Dieses Frequenzspektrum besteht aus der Überlagerung mehrerer Schwingungen oder Wellen mit unterschiedlicher Amplitude und Frequenz beziehungsweise Wellenlänge. Diese Eigenschaften wurden erstmalig 1918 durch Walter Schottky beschrieben. Später wurde das thermische Rauschen experimentell durch John Bertrand Johnson verifiziert. Eine Modellvorstellung der spektralen Leistungsdichte des thermischen Rauschens erfolgte durch Harry Nyquist

Das in diesem Zufallsgenerator verwendete 1/f-Rauschen bezeichnet ein Rauschen, dass mit steigender Frequenz abnimmt, die Amplitudenverteilung ist umgekehrt proportional zur Frequenz (~ 1/f). Die verwendete Rauschquelle ist ein Transistor mit ausgeprägtem Avalanche-Effekt. Rauschen entsteht hier durch den Lawineneffekt (Avalancheeffekt) in der pn-Sperrschicht des Halbleiterbauelements.

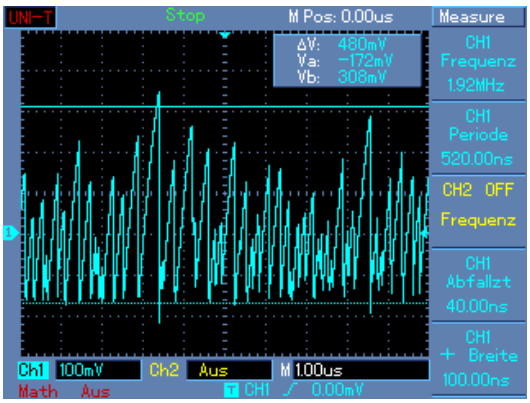


Abbildung: Rauschsignal des eingesetzten Transistors ohne Verstärkung

9 Generierung des Zufallssignals

Der eingesetzte Transistor kann reproduzierbar sehr hohe Rauschspannungen (>300mVss) bei einem breiten Rauschspektrum erzeugen, so dass keine Verstärkung erforderlich ist und der Signalpegel deutlich über dem Störpegel elektronischer Schaltungen liegt. Diese Rauschquelle muss nicht ausgemessen werden, da sie, technologisch bedingt, immer gleiche Rauschamplituden und ein gleiches Frequenzspektrum erzeugt. Ursache des Rauschens ist ein ausgeprägter Avalanche-Effekt.

Zur Digitalisierung des generierten Rauschsignals wird ein integrierter Analog-Komparator des eingesetzten Mikrocontrollers verwendet. Die Referenzspannung wird aus dem Gleichspannungsanteil des Rauschsignals erzeugt.

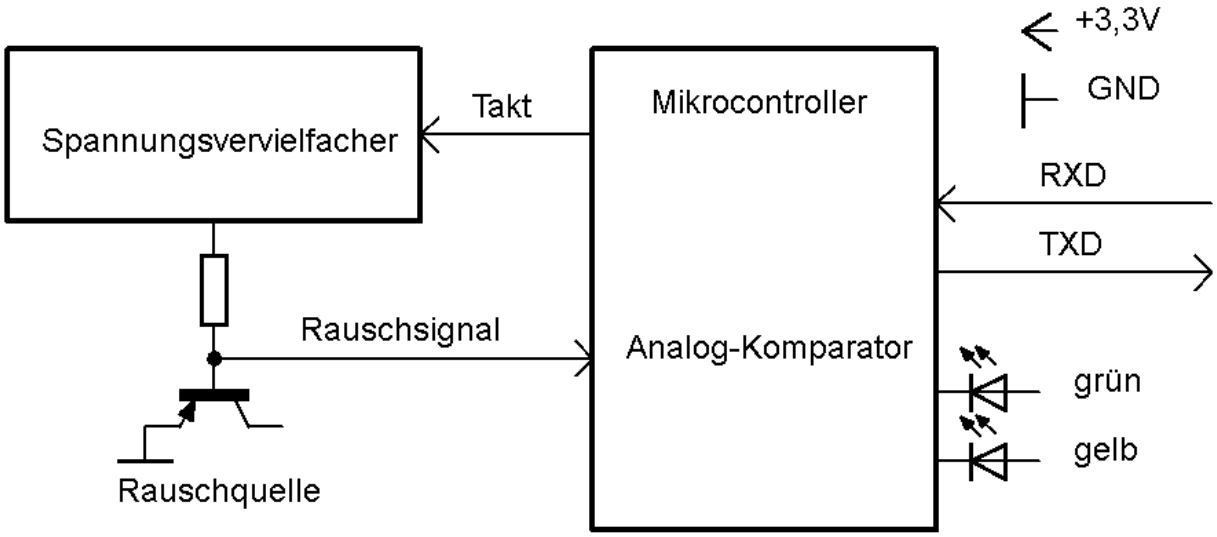


Abbildung: Blockschaltbild des PRG610

Abbildung: analoges und digitalisiertes Rauschsignal

10 Entropie

Die Entropie der Zufallsrohdaten ist die entscheidende Eigenschaft eines Zufallsgenerators und sollte so hoch als möglich sein. Die Generierung von Zufallsrohdaten ist per Kommando möglich. Zur Ermittlung der Entropie wurden jeweils 10Mbyte-Dateien generiert.

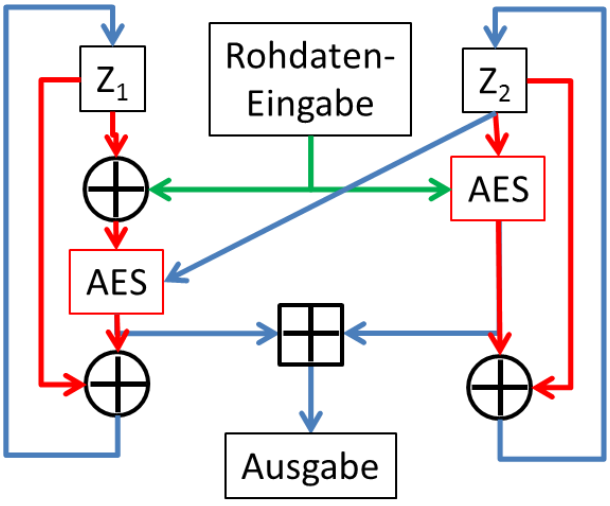
Folgende Entropiewerte (nach Shannon) der Rohdaten wurden für verschiedene Applikationen ermittelt:

PRG610	Mittelwert 0/1	Entropie der Rohdaten
Modul 1	0.50010359	7.99999975
Modul 2	0.49723013	7.99982290
Modul 3	0.49873226	7.99996290
Modul 4	0.49797257	7.99990512
Modul 5	0.49929176	7.99998842

11 Schema der kryptografischen Nachbearbeitung

Auf Grund der sehr hohen und robusten Entropie des physikalischen Zufallsgenerators wurden die Funktionen für die Zufallsgenerierung für die Klasse PTG.3 (hybrider Zufallsgenerator) ausgelegt. Für diese Klasse können beliebig lange Zufallsfolgen generiert werden.

In der Klasse PTG.3 erfolgt die Nachbearbeitung der generierten Rohdaten mit Mayer-Einwegfunktionen mit folgendem Schema (rechtes Bild):



Diese Nachbereitung nutzt 2 Prinzipien des Entropiesammelns: das Aufxorieren der Rohdaten für z1 und die Glättung durch Aufxorieren des mit zufälligem Schlüssel verschlüsselten Zustands für z2. Beide Komponenten nutzen die Mayer-Einwegfunktion AES(k,s) xor k, um den vorangegangenen inneren Zustand zu schützen. Durch die XOR-Summe der Zwischenwerte kann nicht auf den inneren Zustand geschlossen werden.

Dadurch werden auch bei einem kurzzeitigen Ausfall oder Manipulation der Rauschquellen statistisch gleichverteilte Zufallsdaten ausgegeben. Aus 16 Byte Rohdaten werden 16 Byte Zufallszahlen der Klasse PTG.3 generiert. Dieser Prozess kann kontinuierlich oder in definierter Anzahl per Kommando gesteuert werden.

12 Sicherheitsfunktionen

15.1 Tot-Test

Es ist ein Kontrollsystem installiert, welches das digitalisierte Rauschsignal am Anschluss des Mikrocontrollers überwacht. Unmittelbar vor jeder Ausgabe der nach Schema nachbearbeiteten Zufallsrohdaten wird eine Funktion aufgerufen, die folgende Aufgabe hat:

- Es wird die Zeit ermittelt die erforderlich ist, um vier wechselnde Flanken des digitalisierten Rauschsignals zu erfassen
- Wird nach 50µs (entspricht 40 KHz) diese Bedingung nicht erfüllt, wird eine Fehlermeldung generiert, die zur sofortigen Einstellung aller Zufallsausgaben führt. Beide Leuchtdioden leuchten permanent.
- Dieser Zustand ist nur ist nur durch einen automatisch aktivierten „Intensiven Selbsttest" oder PON aufzulösen
- Typische Zeiten, um die Bedingung zu erfüllen, sind 5..8µs

15.2 Permanenter Online-Test

Im permanenten Halbbytetest (zyklisch im Abstand von 1 Sekunde) zur Kontrolle der statistischen Qualität der Zufallsrohdaten werden folgende drei Kriterien differenziert:

- Sind die Werte innerhalb der statistischen Vorgaben, wird kein Fehler generiert
- Sind die Werte außerhalb der statistischen Vorgaben, aber noch innerhalb von weitestgehend gleichverteilten Zufallsdaten, wird ein Fehlerzähler inkrementiert und beide Leuchtdioden blinken im Sekundentakt
- Ist einer der 16 Werte im Halbbytetest gleich Null, wird von einem Totalausfall ausgegangen und jede Zufallsausgabe blockiert. Angezeigt wird dieser Zustand durch das Leuchten der beiden LEDs. Dieser Zustand ist nur durch einen automatisch aktivierten „Intensiven Selbsttest" oder PON aufzulösen.

13 Bedeutung der Leuchtdioden

Die auf der Platine befindlichen Leuchtdioden reflektieren die jeweils ablaufenden Funktionen und Zustände des PRG610. Das Blinken der Leuchtdioden erfolgt immer im Sekundentakt. Synchron zur Änderung des Blinkens (ein→aus und aus→ein) wird der permanente Online-Test gestartet.

LED grün	LED gelb	Zustand
Blinkt	Aus	Selbsttest ok, es wird keine Funktion ausgeführt
Blinkt	Ein	Selbsttest ok, eine Funktion wird ausgeführt
Blinkt	Blinkt	Statistischer Fehler im Online-Test
Ein	Ein	Hardwarefehler im Online- oder Tot-Test festgestellt

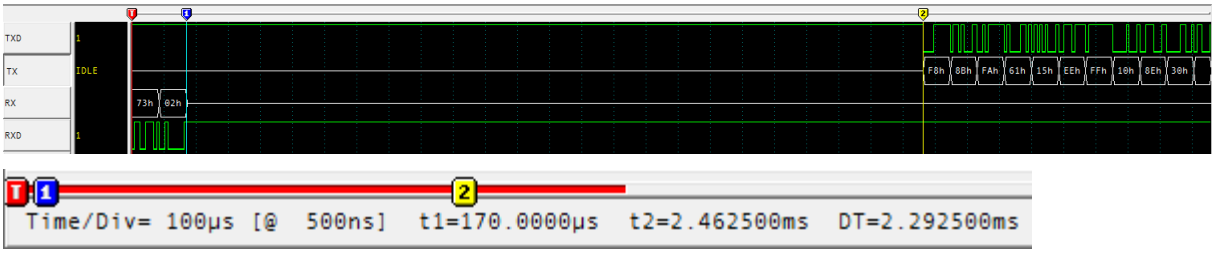
14 Signaturanalyse

Im Folgenden werden Signaturanalysen der Datenausgabe und ausgewählter Sicherheitsfunktionen demonstriert. Dazu war es notwendig, in der Entwicklungs-Firmware Zusatzsignale zu generieren, um eindeutige Signaturen zu demonstrieren.

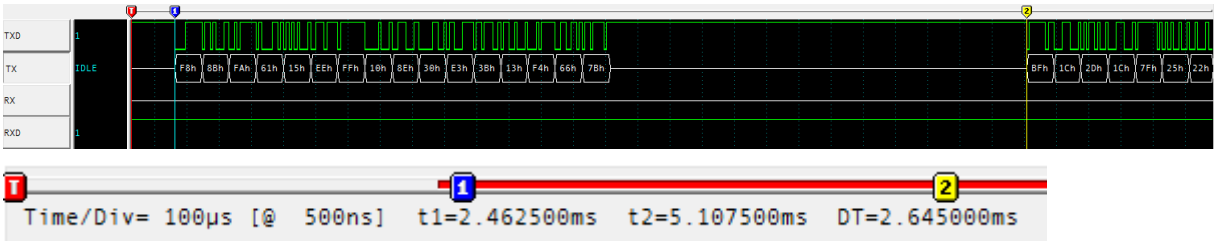
12.1 Datenausgabe

Beispiel Generierung von 32 Byte Zufall nach PTG.3

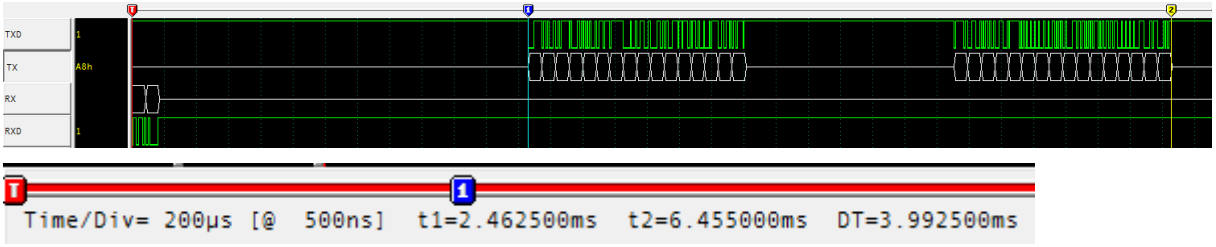
Zeit vom Kommando bis zur Datenausgabe: 2,29ms



Zeit zwischen den Ausgabeblöcken (jeweils 16 Byte): 2,64ms (entspricht 48 Kbit/s)

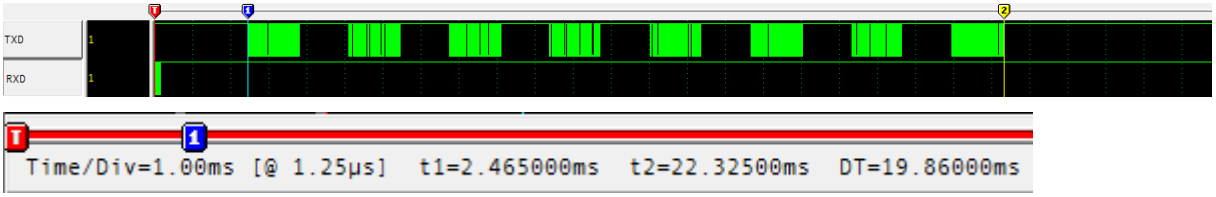


Gesamtzeit der Abarbeitung: 6,45ms



Beispiel Generierung von 128 Byte Zufall nach PTG.3

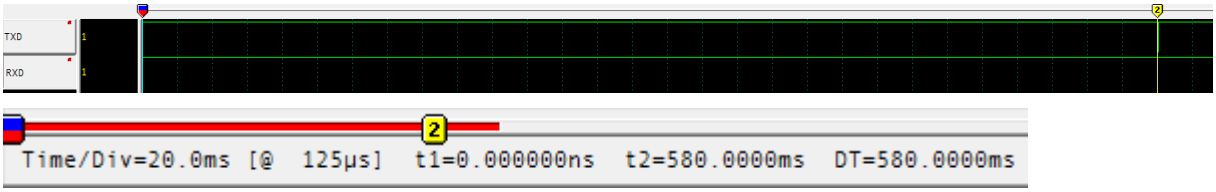
Gesamtzeit der Abarbeitung: 22,32ms



12.1 Sicherheitsfunktionen

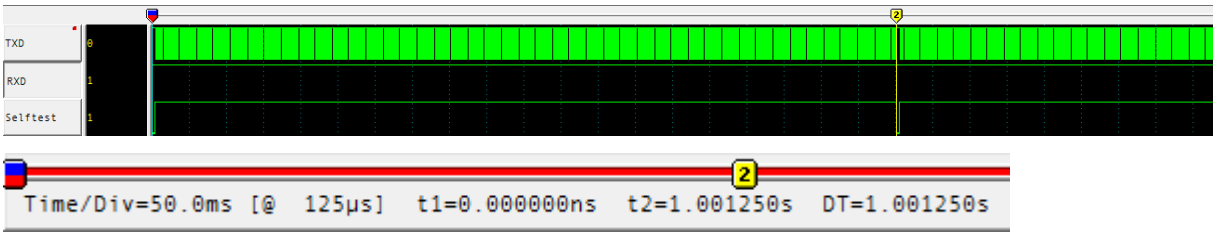
Intensiver Selbsttest

Zeit für einen Durchlauf (Start bis Rückkehr-Code) des intensiven Selbsttest: 580ms



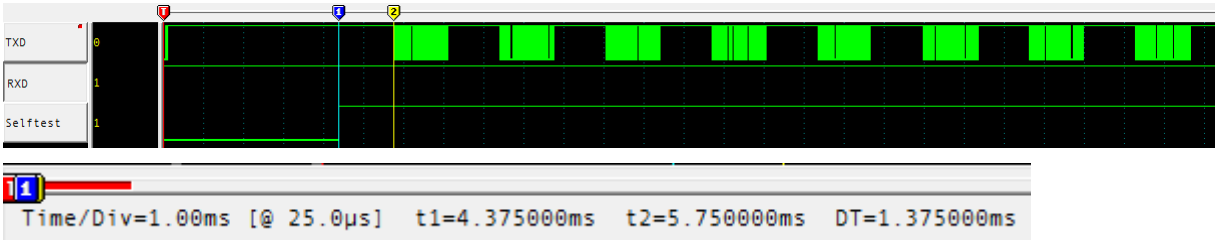
Permanenter Selbsttest (Online-Test)

Abstand zwischen zwei Selbsttest: 1,0s



Permanenter Selbsttest (Online-Test)

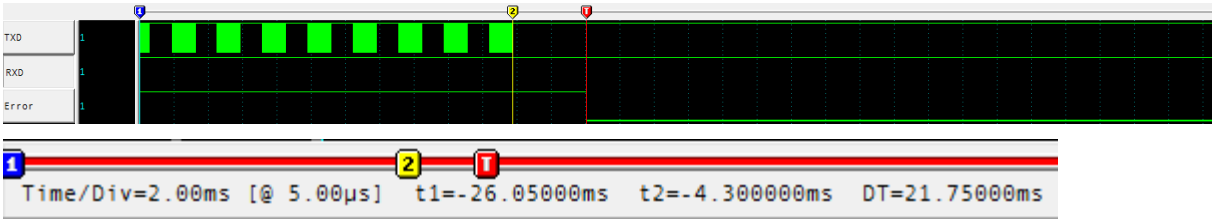
Zeit für einen Permanenten Selbsttest: 4,37ms



12.2 Fehlermeldung

Simuliert wird der Ausfall des Rauschsignals: das ERROR-Signal wird nur im Mikrocontroller generiert. Es folgt ein automatischer „Intensiver Selbsttest" solange, bis die internen statistischen Analysen wieder fehlerfrei sind.

Reaktionszeit zwischen letztem Zufallsblock und Blockierung: 4,3ms



15 Statistische Qualität

Dieser physikalische Zufallsgenerator generiert kontinuierlich Zufallsbits in herausragender statistischer Qualität. Eine Qualitätsaussage zu einem solchen Produkt ist aber nicht durch einen einzelnen statistischen Test möglich. In Zusammenarbeit mit Mathematikern und Kryptologen hat sich die Summe aus folgenden statistischen Tests für eine sichere Qualitätsaussage eines physikalischen Zufallsgenerators bewährt:

- Statistische Forderungen der AIS31-Dokumente für Rohdaten (keine digitale Nachbearbeitung, denn nichts beschreibt besser die Eigenschaften eines physikalischen Zufallsgenerators, als seine Rohdaten!)
- NIST-Test-Suite (Summe verschiedener Test der USA-Sicherheitsbehörde)
- Diehard-Test-Suite nach George Marsaglia
- Proprietärer statistischer Basistest

Zur Evaluierung wurden umfangreiche statistische Tests durchgeführt. So wurden die ausgewählten Tests auf mehrere erzeugte Bitfolgen angewendet. Keine dieser Testfolgen konnte Unterschiede zu einem idealen Zufallszahlengenerator aufzeigen. Darüber hinaus wurden Testergebnisse mit Untersuchungen von kryptographisch starken Pseudo-Zufallszahlengeneratoren, wie dem DES-, AES- und SH1-Generator verglichen. Die Vergleiche zeigten keine signifikanten Unterschiede zu den Testergebnissen dieser deterministischen Generatoren.

16 Anwendungen

Auf Grund der sehr hohen Entropie des physikalischen Zufallsgenerators wurden die Funktion für die Klasse PTG.3 (hybrider Zufallsgenerator) ausgelegt. Bezugnehmend auf die AIS31-Dokumente (Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren: www.bsi.bund.de/zertifiz/zert/interpr/ais31e.pdf) ist ein permanenter Online-Test und ein Tot-Test (Ausfall der Rauschquelle) der Zufallsrohdaten integriert. Fällt die Rauschquelle aus oder werden statistische Grenzen überschritten, wird die Ausgabe von Zufallsdaten so lange blockiert, bis alle Parameter wieder im Toleranzbereich liegen.

Der stetig steigende Bedarf an Zufallszahlen in vielen Bereichen von Wissenschaft und Technik erfordert eine zuverlässige und stabile Generierung von Zufallszahlen mit physikalischem Zufall und sicherer Gewährleistung aller erforderlichen statistischen und funktionellen Normen. Damit sind vor allem Entwickler von Sicherheitsapplikationen in der Lage, Wirksamkeit und Widerstand gegen Angriffe besser zu kalkulieren. Besonders hohe Ansprüche an die Statistik von Zufallszahlen werden für kryptografisch sichere Zufallszahlen gestellt.

Exemplarische Beispiele für den Einsatz des PRG610:

- Implementierung in proprietäre Applikationen der IT-Sicherheit
- einfachere Administration und sichere Verschlüsselung bei drahtloser Datenübertragung: WLAN, Bluetooth, GSM, ZiggBee, Industriedatenfunk
- für die Generierung kryptografisch sicherer Parameter auf Bords wie:
    - Raspberry-Pi
    - Odroid-C1
    - Banana-Pi

17 Einsatzumgebung

Der PRG610 ist für den permanenten Einsatz in beliebigen Applikationen entwickelt und getestet worden. Auch bei erhöhtem Industriestandard (0°C bis +70°C) bleiben die Entropiewerte sehr hoch und unterschreiten die Vorgaben aus den AIS31-Dokumenten nicht. Statistische Analysen, auch für diese Temperaturbereiche, befinden sich auf der Internetseite des Autors.

17 Kommando-Interface

17.1 Versionsabfrage

Übergabeparameter

0x76

RC:

String mit Version, String-Ende mit 0x0a, 0x0d

17.2 Intensiver Selbsttest

Statistisch bewertet können 1% aller Tests negativ sein.

Übergabeparameter

0x74

RC:

1. Byte	0x55 ok
	0xaa Qualität der Zufallsrohdaten nicht ausreichend
2. Byte	Anzahl der benötigten Versuche (0x05 = ein Versuch, 0x04 = zwei Versuche,..0x01 = fünf Versuche, 0x00 = Fehler)

17.3 Abfrage des Fehlerzählers

Werden beim zyklischen Selbsttest die statistischen Grenzen über- oder unterschritten, wird der Fehlerzähler inkrementiert und kann jederzeit abgefragt werden. Wird der „Intensive Selbsttest" erfolgreich durchlaufen, wird der Fehlerzähler wieder gelöscht, ebenso nach PON. Hat der Fehlerzähler seinen Maximalwert erreicht, wird automatisch ein interner, intensiver Selbsttest aktiviert.

Übergabeparameter

0x66

RC:

1 Byte Fehlerzähler (0x00..0xff)

17.4 PTG.2: keine Nachbearbeitung der Zufallsrohdaten

Übergabeparameter

0x72

RC:

0x72

17.5 PTG.3: Nachbearbeitung der Zufallsrohdaten nach Schema

Diese Funktion wird automatisch nach PON und nach Beendigung jeder Ausgabe von Zufallsrohdaten eingestellt. Damit wird die Ausgabe kryptografisch sicherer Zufallszahlen gesichert.

Übergabeparameter

0x34

RC:

0x34

17.6 Start der permanenten Zufallsgenerierung

Es wird mit der permanenten Zufallsgenerierung begonnen. **Beendet wird diese Funktion mit Ausgabe des Zeichens 0x65.**

Übergabeparameter

0x62

RC:

Keinen

Permanente Zufallsgenerierung der Klasse PTG.2 oder PTG.3

Beenden: 0x65

17.7 PTG.3: Ausgabe einer definierten Anzahl von Zufallsdaten

Mittels eines übergebenen Parameters können ein Vielfaches von 16 Byte Blöcken erzeugt und ausgegeben werden. Diese Funktion wird sofort gestartet und nach Abarbeitung der gewünschten Anzahl der 16-Byte-Blöcke automatisch beendet.

Übergabeparameter

0x73

1 Byte Anzahl der Ausgabebyte *16 (Wertebereich 0x01..0xff)

RC:

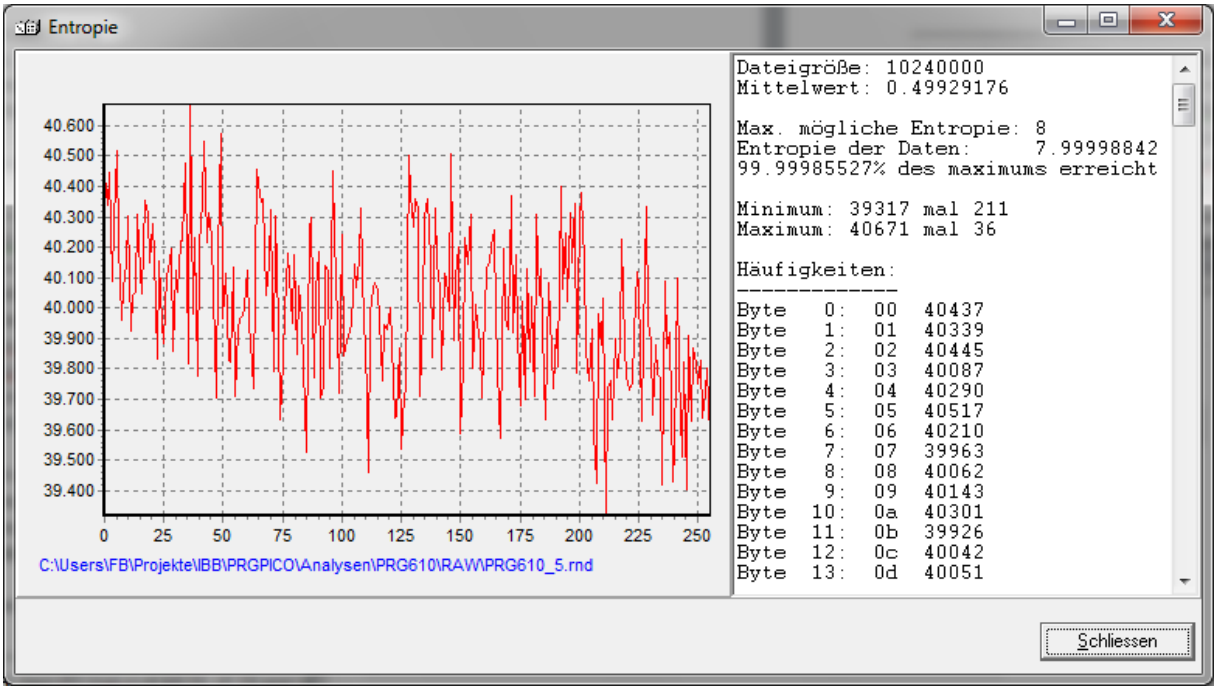
Je nach übergebenem Parameter 16..4080 Byte Zufallsdaten

18 Statistische Analysen (Beispiele)

18.1 Analyse von Rohdaten

Mittelwert: 0.49929176

Entropie der Daten: 7.99998842




```
*******************************************************************************
 *                                                 *
 *                AIS31 evaluation tests                 *
 *                                                 *
*******************************************************************************

 date, time:    01/22/2015,  14:25:35
 tested file:   prg610_5.rnd
 size of file:  10240000 bytes

 ---------------------------------------------------------------
 Results of test 1  (test (P2.i)(vii.a) of AIS 31, cf. [3] and [4])
 ---------------------------------------------------------------

 test scope:  first 100000 bits
 number of ones:  50252
 relative frequency:  0.502520
 test value:  0.00252000  < 0.025

 sequence passes test 1

 ---------------------------------------------------------------
 Results of test 2  (test (P2.i)(vii.b) of AIS 31, cf. [3] and [4])
 ---------------------------------------------------------------

 number of 2-bit words looked up: 200099
```

```
 relative frequency v(0,1):  0.500170
 relative frequency v(1,0):  0.501590
 test value:  0.00176000 < 0.02

 sequence passes test 2

 ------------------------------------------------------------------
 Results of test 3  (test (P2.i)(vii.c) of AIS 31, cf. [3] and [4])
 ------------------------------------------------------------------

 number of 3-bit words looked up: 400687
 test value t_1:    0.968048 <= 15.13
 test value t_2:    0.288001 <= 15.13

 sequence passes test 3

 ------------------------------------------------------------------
 Results of test 4  (test (P2.i)(vii.d) of AIS 31, cf. [3] and [4])
 ------------------------------------------------------------------

 number of 4-bit words looked up: 801604
 test value t_00:    0.131220 <= 15.13
 test value t_01:    0.950486 <= 15.13
 test value t_10:    3.329298 <= 15.13
 test value t_11:    1.310725 <= 15.13

 sequence passes test 4

 ------------------------------------------------------------------
 Results of test 5  (test (P2.i)(vii.e) of AIS 31, cf. [3] and [4])
 ------------------------------------------------------------------

 8-bit words looked up: 2560 + 256000 bytes
 f-value:  8.00117140
 8.00117140 > 7.976

 sequence passes test 5
```

17.1 Analyse von Zufallsdaten der Klasse PTG.3

```
Basic statistical test of bit sequences
=======================================
Date/Time: 26.01.2015,16:21 hour

file: PRG610_5.rnd  size:    10240000 Bytes

Test of null-hypothesis:
------------------------
Bit stream ist a stream of truly randomly
drawn number 0,1 with same probability p = 0.5

Non-overlapping byte count:
```

```
00     39858  40102  40185  40194  39702  39788  39952  40009
08     39613  40130  39855  39915  40061  39888  40020  39867
10     40176  39888  40191  39944  39994  39754  40238  40046
18     40011  39627  39984  40039  40205  40111  39814  39717
20     39925  40004  39926  40075  39680  40377  39928  40120
28     40254  39979  40257  40156  39830  40140  39952  40065
30     40234  40239  39874  40142  40043  39885  40111  39694
38     40029  39820  40225  40020  39650  39920  39804  40209
40     40053  40174  40294  40157  40092  39892  39817  40082
48     39961  39757  39745  40386  39985  40248  40220  40288
50     40103  40032  39818  39869  40013  40238  39782  39997
58     39723  40164  39843  40009  39698  40239  39889  39719
60     40102  40102  39885  39978  40350  40030  40034  40086
68     40238  40119  40008  40031  39920  39747  39893  39972
70     40140  39579  40194  39582  39984  39934  40021  39769
78     39875  39676  39953  40243  39965  40096  39926  40057
80     40041  40299  39607  39714  39982  39776  40339  39487
88     40174  40220  39657  39909  40097  39971  39667  39642
90     39933  39861  40218  39783  40521  40406  39964  39860
98     39932  39970  40334  39969  40138  40325  40239  39863
a0     39746  40021  39897  40193  40136  39943  40325  39620
a8     40252  40145  39960  39850  39999  39734  40368  40157
b0     39846  40016  39768  40232  40167  40491  40087  40021
b8     40024  39865  40118  40074  39739  39750  39851  40105
c0     40026  40499  39970  39594  40208  39802  40351  40231
c8     39946  39902  39909  39915  40063  40065  40075  40073
d0     39640  40022  40036  40194  40214  39987  39988  40306
d8     39765  39586  40461  40049  39702  40186  39704  39761
e0     40000  39854  40073  40279  39782  40037  39800  39865
e8     40164  40005  40033  39955  39709  40042  39973  40051
f0     40243  40017  40044  40255  40145  40261  39838  40082
f8     40239  39952  40046  39899  39555  40031  40131  39791

Evaluation of count of   10240000 Bytes =    81920000 Bits:

  Theoretical average of byte-frequencies: 40000
  '87' = 39487 (minimum)    '94' = 40521 (maximum)

  Theoretical interval I of byte-frequencies:
  I = (39609 to 40391) (for 95 % of 256 frequency)

Test 1:
  The theoretical permissible number of the 5% outliers (average 13)
  from the interval I is between 6 and 20

  The real number of the outliers from interval I:
  smaller: 7  greater: 5  summary: 12

Test 2:
  Evaluation of byte-frequencies
  Chi-square non-overlapping:
  Theoretical maximum chi-square = 293.25
  Chi-square value             = 258.05

  Chi-square overlapping:
  Theoretical maximum chi-square = 155.40
  Chi-square value             = 151.31
```

```
Test 3:
  r = 0.49996781 (relative frequency of bit 1 in the bit stream)

  For a truly random sequence, the probability for r to have values in the
  complement of the open interval (0.49996781 , 0.50003219) is w = 0.56013691.
  If w is very small (e.g., w < 0.05), the null-hypothesis is rejected.
  If more sequences can be tested, the probability w has to be >= 0.05
  for about 95% of the tested bit sequences.

Test 4:
  Frequencies of overlapping 2-tuples:
  tuples 00:   20479470    tuples 01:   20483166
  tuples 10:   20483166    tuples 11:   20474198

  Check size: Chi-square of 2-bit patterns minus chi square of 1-bit patterns
  Theoretical maximum chi-square = 5.99
  Chi-square value            = 2.30

Test 5:
  Frequencies of 2-tuples on even places:
  tuples 00:   10239481    tuples 01:   10241752
  tuples 10:   10241922    tuples 11:   10236845

  Theoretical maximum chi-square = 7.81
  Chi-square value            = 1.66

Test 6:
  Frequencies of 2-tuples on odd places:
  tuples 00:   10239989    tuples 01:   10241414
  tuples 10:   10241244    tuples 11:   10237353

  Theoretical maximum chi-square = 7.81
  Chi-square value            = 1.03

Result of statistical analysis of file PRG610_5.rnd:
================================================
The tests: 1 2 3 4 5 6  were fulfilled!

The null-hypothesis is accepted!



NIST-Test-Suite
############################################################
          THE NIST STATISTICAL TEST SUITE
############################################################
------------------------------------------------------------
  1.  FREQUENCY TEST
------------------------------------------------------------
Computational information:
(a) The nth partial sum = -158
(b) S_n/n           = -0.000158

p_value = 0.874457, SUCCESS
```

```
---------------------------------------------------------
  2.  BLOCK FREQUENCY TEST
---------------------------------------------------------
Computational information:
(a) Chi^2          = 125456.500000
(b) # of substrings = 125000
(c) block length    = 8

p_value = 0.180562, SUCCESS

---------------------------------------------------------
  3.  CUMULATIVE SUMS TEST
---------------------------------------------------------
Cumulative sums forward test:
Computational information:
(a) The maximum partial sum =

p_value = 0.759852, SUCCESS

Cumulative sums reverse test:
Computational information:
(a) The maximum partial sum =

p_value = 0.754257, SUCCESS

---------------------------------------------------------
  4.  RUNS TEST
---------------------------------------------------------
Computational information:
(a) Pi                    = 0.499921
(b) V_n_obs (Total # of runs) = 500309
(c) V_n_obs - 2 n pi (1-pi)
    ---------------------- = 0.437010
     2 sqrt(2n) pi (1-pi)

p_value = 0.536559, SUCCESS

---------------------------------------------------------
  5.  LONGEST RUNS OF ONES TEST
---------------------------------------------------------
Computational information:
(a) N (# of substrings)  = 100
(b) M (Substring Length) = 10000
(c) Chi^2              = 4.478808
```

```
Frequency
-----------------------------------------------
<=10    11    12    13    14    15   >=16
-----------------------------------------------
 12    22    22    18    15     3     8

p_value = 0.612168, SUCCESS

------------------------------------------------------------
 6.  RANK TEST
------------------------------------------------------------
Computational information:
(a) Probability P_32 = 0.288788
(b)           P_31 = 0.577576
(c)           P_30 = 0.133636
(d) Frequency   F_32 = 291
(e)           F_31 = 572
(f)           F_30 = 113
(g) # of matrices    = 976
(h) Chi^2           = 2.747229
(i) NOTE: 576 BITS WERE DISCARDED.

p_value = 0.253190, SUCCESS

------------------------------------------------------------
 7.  DFT TEST
------------------------------------------------------------
Computational information:
(a) Percentile = 95.001400
(b) N_l        = 475007.000000
(c) N_o        = 475000.000000
(d) d          = 0.045422

p_value = 0.963771, SUCCESS

------------------------------------------------------------
 8.  NONOVERLAPPING TEMPLATES TEST
------------------------------------------------------------
Computational information:
LAMBDA = 122.061523
M = 125000, N = 8, m = 10, n = 1000000
Template   W_1  W_2  W_3  W_4  W_5  W_6  W_7  W_8
-----------------------------------------------
1100100100 107  128  118  125  128  111  136  123
chi2_value =  5.342252

p_value = 0.720447, SUCCESS
```

---
 9. OVERLAPPING TEMPLATE OF ALL ONES TEST

Computational information:
(a) n (sequence_length)   = 1000000
(b) m (block length of 1s)   = 10
(c) M (length of substring) = 1032
(d) N (number of substrings) = 968
(e) lambda [(M-m+1)/2^m]   = 0.999023
(f) eta   = 0.499512
Frequency:
---
0   1   2   3   4   >=5   Chi^2
---
607 140   99   48   24   50   6.0360

p_value = 0.302731, SUCCESS

---
 10. UNIVERSAL TEST

Computational information:
(a) L   = 7
(b) Q   = 1280
(c) K   = 141577
(d) sum   = 876908.163070
(e) sigma   = 0.002768
(f) variance = 3.125000
(g) exp_value = 6.196251
(h) phi   = 6.193860
(i) WARNING: 1 bits were discarded.

p_value = 0.387895, SUCCESS

---
 11. APPROXIMATE ENTROPY TEST

Computational information:
(a) m (block length)   = 5
(b) n (sequence length) = 1000000
(c) Chi^2   = 27.503392
(d) Phi(m)   = -3.465727
(e) Phi(m+1)   = -4.158861
(f) ApEn   = 0.693133
(g) Log(2)   = 0.693147

p_value = 0.693687, SUCCESS

---
12. RANDOM EXCURSIONS TEST

Computational information:
(a) Number Of Cycles (J) = 1774
(b) Sequence Length (n) = 1000000
(c) Rejection Constraint = 500.000000

x = -4 chi^2 = 1.749357 p_value = 0.882624, SUCCESS
x = -3 chi^2 = 2.818955 p_value = 0.727873, SUCCESS
x = -2 chi^2 = 1.591243 p_value = 0.902306, SUCCESS
x = -1 chi^2 = 3.461105 p_value = 0.629281, SUCCESS
x = 1 chi^2 = 4.655017 p_value = 0.459414, SUCCESS
x = 2 chi^2 = 5.241221 p_value = 0.387156, SUCCESS
x = 3 chi^2 = 0.828632 p_value = 0.975171, SUCCESS
x = 4 chi^2 = 8.047289 p_value = 0.153649, SUCCESS

---
13. RANDOM EXCURSIONS VARIANT TEST

Computational information:
(a) Number Of Cycles (J) = 1774
(b) Sequence Length (n) = 1000000

(x = -9) Total visits = 1513; p-value = 0.287903 SUCCESS
(x = -8) Total visits = 1603; p-value = 0.458548 SUCCESS
(x = -7) Total visits = 1613; p-value = 0.453462 SUCCESS
(x = -6) Total visits = 1600; p-value = 0.378444 SUCCESS
(x = -5) Total visits = 1557; p-value = 0.224611 SUCCESS
(x = -4) Total visits = 1574; p-value = 0.204412 SUCCESS
(x = -3) Total visits = 1601; p-value = 0.193985 SUCCESS
(x = -2) Total visits = 1642; p-value = 0.200741 SUCCESS
(x = -1) Total visits = 1733; p-value = 0.491250 SUCCESS
(x = 1) Total visits = 1818; p-value = 0.460097 SUCCESS
(x = 2) Total visits = 1790; p-value = 0.876755 SUCCESS
(x = 3) Total visits = 1757; p-value = 0.898437 SUCCESS
(x = 4) Total visits = 1726; p-value = 0.760687 SUCCESS
(x = 5) Total visits = 1608; p-value = 0.352912 SUCCESS
(x = 6) Total visits = 1496; p-value = 0.159367 SUCCESS
(x = 7) Total visits = 1444; p-value = 0.124400 SUCCESS
(x = 8) Total visits = 1439; p-value = 0.146464 SUCCESS
(x = 9) Total visits = 1427; p-value = 0.157683 SUCCESS

---
14. SERIAL TEST

Computational information:
(a) Block length (m) = 5
(b) Sequence length (n) = 1000000
(c) Psi_m = 17.047040
(d) Psi_m-1 = 3.653440
(e) Psi_m-2 = 1.458224
(f) Del_1 = 13.393600
(g) Del_2 = 11.198384

p_value1 = 0.643791, SUCCESS
p_value2 = 0.190710, SUCCESS

---
15. LEMPEL-ZIV COMPRESSION TEST

Computational information:
(a) W (# of words) = 69600

p_value = 0.915995, SUCCESS

18 Literatur

[1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model; CCMB-2012-09-001, Version 3.1, Revision 4, September 2012

[2] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components; CCMB-2012-09-002, Version 3.1, Revision 4, September 2012

[3] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements; CCMB-2012-09-003, Version 3.1, Revision 4, September 2012

[4] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology; CCMB-2012-09-004, Version 3.1, Revision 4, September 2012

[5] AIS20: Functionality classes and evaluation methodology for deterministic random number generators, Version 2.1, 02.12.2011, Bundesamt für Sicherheit in der Informationstechnik

[6] AIS31: Functionality classes and evaluation methodology for true (physical) random number generators, Version 2.1, 02.12.2011, Bundesamt für Sicherheit in der Informationstechnik

[7] Killmann, W. Schindler, „A proposal for: Functionality classes for random number generators“, Version 2.0, September 18, 2011

[8] Evaluation of random number generators, Version 0.8, Bundesamt für Sicherheit in der Informationstechnik