

Ръководство за експлоатация

GTR-01

Квантов двуканален 2.2 Mb/s
USB генератор на случайни числа

Рев. 2.4 – 02 май 2025



BENEV®

Продуктовите спецификации описани в настоящия документ са обект на промяна без предварително известяване. Софтуерът и документацията са с пълни авторски права притежание на *Бенев наука и технология ЕООД*. Никаква част от информацията съдържаща се в този документ или от продуктите описани в него не може да бъде модифицирана или възпроизвеждана, под каквато и да е материална или електронна форма, без изричното писмено съгласие от притежателя на авторските права.

Copyright © 2025 Бенев наука и технология ЕООД. Всички права запазени.

BENEV® и *StudioGTR™* са търговски марки или регистрирани търговски марки от *Бенев наука и технология ЕООД*. Всички други марки или продуктови наименования, споменати в този документ, са търговски марки или регистрирани търговски марки от съответните им притежатели.




Бенев наука и технология ЕООД

Тел: +359 89 923 0345


Уебсайт: www.BenevSciTech.com


бул. „Цар Симеон Велики“ 141
Стара Загора 6000,
България

Използвани означения

Символ	Значение
	Действие свързано с работата на софтуера
 БЕЛЕЖКА:	Допълнителна информация
 ВНИМАНИЕ:	Информация с висока степен на важност

Безопасност

 **БЕЛЕЖКА:** Преди да използвате GTR-01 се запознайте с настоящото ръководство за експлоатация.

 **ВНИМАНИЕ:** Не излагайте устройството на въздействия извън границите на максимално допустимите, описани в документацията! Използвайте уреда само по предназначение и съблюдавайте стриктно указанията за употреба!

Гаранция

Бенев наука и технология ЕООД гарантира, че този продукт ще работи без дефекти по отношение както на материалите и компонентите използвани при производството му, така и по отношение на цялостната му изработка, за срок от 2 (две) години от датата на закупуването му. Ако този продукт покаже дефект по време на гаранционния си период, *Бенев наука и технология ЕООД* поема отговорността за поправката или цялостната замяна на продукта.

Задължение на клиента е да ни уведоми при възникване на дефект в рамките на гаранционния период, както и грижата по опаковането и поемането на транспортните разходи до нашия офис. *Бенев наука и технология ЕООД* поема транспортните разходи по доставянето на ремонтирания или заменен продукт обратно до клиента.

Гаранцията не покрива дефекти и повреди в следствие от неправилни употреба и съхранение на продукта. *Бенев наука и технология ЕООД* няма задължение по изпълнение на гаранцията в следните случаи:

- Този продукт е бил разглобяван, модифициран или поправян от лица, които не са били изрично упълномощени от *Бенев наука и технология ЕООД* за това, както и в случай на увреждане или подмяна на гаранционните стикери.

- Този продукт е бил подлаган на въздействия извън границите на допустимост посочени в документацията му или е бил употребяван в съчетание с неизправни или с неотговарящи на необходимите норми за качество и безопасност продукти.

- Възникване на форсмажорни обстоятелства – пожар, земетресение, наводнение и др.

Съдържание

Използвани означения	iii
Безопасност	iv
Гаранция	v
Списък на фигурите	vii
Списък на таблиците	vii
1. Начало	1
1.1 Основни особености	1
1.2 Приложения	1
2. GTR-01	2
2.1 Общ вид на устройството	2
2.2 Процес на генериране на шум	3
2.3 Принципна схема	4
2.4 Технически спецификации	4
2.5 Инсталиране на драйверите	5
3. StudioGTR	8
3.1 Системни изисквания	8
3.2 Инсталиране на програмата	8
3.3 Главен екран	9
3.4 Общ преглед на възможностите на програмата	9
3.5 Инструменти	11
3.5.1 Визуализатор	11
3.5.2 Цели числа	12
3.5.3 Реални числа	13
3.5.4 Символни низове	14
3.5.5 Случаен шум	16
3.5.6 Изображения	17
3.5.7 Лотарийни числа	18
3.5.8 Регистратор случайни събития	19
3.5.9 Статистически тестове	20
3.5.10 Калибриране	21
3.6 Настройки	22
3.6.1 Канал	22
3.6.2 Трансфер	22
3.6.3 Псевдослучаен генератор	23
3.7 Многоезичност	24
3.8 Предпочитания за програмата	25

4. API на ниско ниво	26
4.1 GTR-01 IO команди	26
4.1.1 GET_RND_CHANNEL_A	27
4.1.2 GET_RND_CHANNEL_A_CALIBRATION	28
4.1.3 GET_RND_CHANNEL_B	28
4.1.4 GET_RND_CHANNEL_B_CALIBRATION	28
4.1.5 GET_RND_NORMAL	28
4.1.6 GET_RND_NORMAL_PLUS	29
4.1.7 GET_RND_NORMAL_DEBIAS	29
4.1.8 GET_RND_CHANNEL_A_ASYNC	29
4.1.9 GET_RND_CHANNEL_A_CALIBRATION_ASYNC	29
4.1.10 GET_RND_CHANNEL_B_ASYNC	30
4.1.11 GET_RND_CHANNEL_B_CALIBRATION_ASYNC	30
4.1.12 GET_RND_NORMAL_ASYNC	30
4.1.13 GET_RND_NORMAL_PLUS_ASYNC	30
4.1.14 GET_RND_NORMAL_DEBIAS_ASYNC	31
4.1.15 SET_CALIBRATION	31
4.1.16 GET_CALIBRATION	31
4.1.17 GET_DEVICE_ID	32
Приложение А: Статистически тестове	33
Приложение В: FTDI IO примерен код на C#	34
Приложение С: GTR-01 IO пример	37
Азбучен указател	40

Списък на фигурите

2.1	GTR-01 (вид отгоре)	2
2.2	GTR-01 преден панел	2
2.3	Обратно свързан p-n преход	3
2.4	Волт-амперна характеристика на p-n преход	3
2.5	GTR-01 принципна схема	4
2.6	Инсталиране на драйверите – стъпка 1	5
2.7	Инсталиране на драйверите – стъпка 2	5
2.8	Инсталиране на драйверите – стъпка 3	6
2.9	Инсталиране на драйверите – стъпка 4	6
2.10	Инсталиране на драйверите – стъпка 5	6
2.11	Инсталиране на драйверите – стъпка 6	6
2.12	Успешно инсталиране на драйверите	7
3.1	StudioGTR главен екран	9
3.2	Визуализатор – екран	11
3.3	Цели числа – екран	12
3.4	Реални числа – екран	13
3.5	Символни низове – екран	14
3.6	Случаен шум – екран	16
3.7	Изображения – екран	17
3.8	Лотарийни числа – екран	18
3.9	Регистратор случайни събития – екран	19
3.10	Статистически тестове – екран	20
3.11	Калибриране – екран	21
3.12	Синхронен режим на работа	22
3.13	Асинхронен режим на работа	23
3.14	Диалогов прозорец с предпочитания	25

Списък на таблиците

1.1	Основни особености	1
2.1	Технически спецификации	4
3.1	Системни изисквания	8
4.1	Списък на IO команди	27
A.1	Статистически тестове	33

1. Начало

GTR-01 е хардуерна система за генериране на истински случайни числа. Иновативният мултиплатформен дизайн, удобният USB интерфейс, високата скорост на трансфер от 2.2 Mb/s и компактният размер на GTR-01, превръщат този инструмент в най-правилният избор за много приложения, свързани с криптография и сигурност, научни изследвания, компютърни симулации, моделиране, лотарии, хазарт, приложна математика и статистика и т.н. В много от горните области е важно да се използва източник на случайни числа, базиран на реален физичен феномен, за разлика от компютърно генерираните псевдо случайни числа, използващи известен и предвидим числов алгоритъм.

GTR-01 е разработен с визия към съвременния хардуерен дизайн, осигуряващ сигурност и функционална надеждност от една страна и достатъчно опростен и оптимизиран софтуерен интерфейс за високоскоростна комуникация с компютъра, от друга. Работата на GTR-01 се основава на физическо явление, наречено лавинен шум в обратно свързан p-n преход и е квантово по своята природа. Това го прави подходящ избор за източник на ентропия в хардуерните генератори на случайни числа.

1.1 Основни особености

Особености
Квантов процес като източник на ентропия
Два напълно независими аналогови канала с XOR функция
12-битова прецизна калибровка със запис на коефициентите в EEPROM
Скорост на трансфер до 2.2 Mb/s случайни бита през стандартен USB интерфейс
Многоплатформен дизайн включващ Windows, Linux, Mac OS и Android
API интерфейс включващ графично приложение и примерен код
Качество на изходните данни проверено чрез статистически тестове
Компактна и здрава, екранирана алуминиева кутия

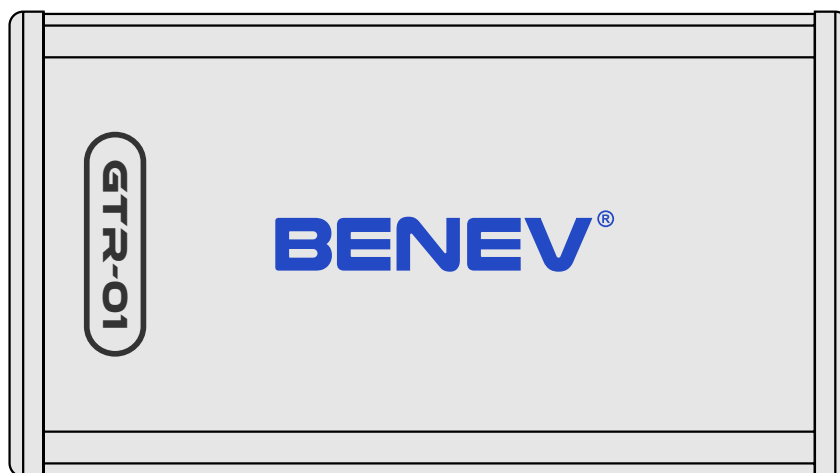
1.2 Приложения

- Криптография и сигурност
- Лотарии и хазартни игри
- Генериране на ПИН кодове
- Научни изследвания
- Компютърни симулации
- Статистика

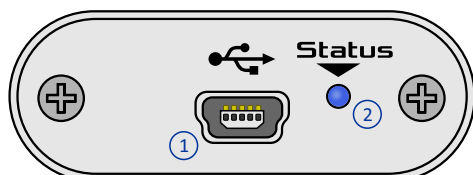
2. GTR-01

2.1 Общ вид на устройството

GTR-01 (изглед отгоре)



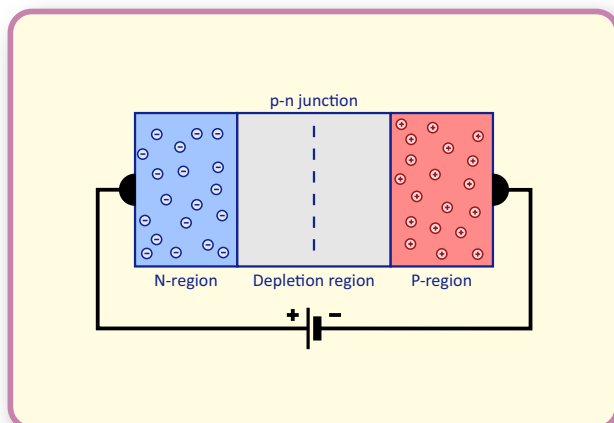
GTR-01 преден панел



1. USB mini конектор
2. Статусна LED индикация

2.2 Процес на генериране на шум

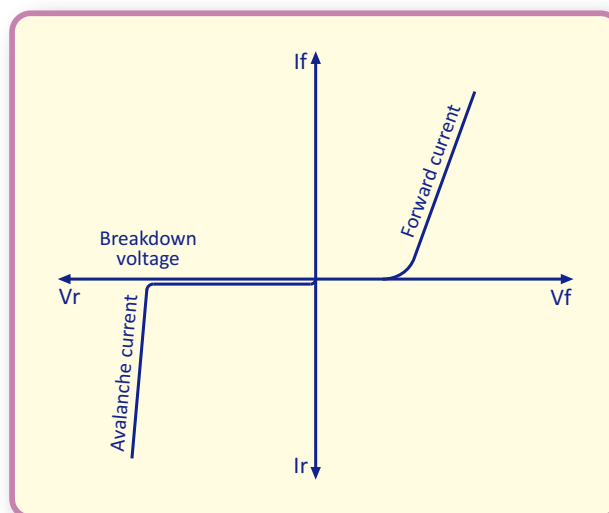
GTR-01 използва физическо явление, известно като лавинен шум в обратно свързан р-п преход, за основен източник на ентропия.



Обратно свързан р-п преход.

При подаване на обратно напрежение (*reverse bias*), областта на обедняване (*depletion region*) на р-п прехода се разширява. По този начин през прехода възниква достатъчно силно електрично поле. Освободените поради случайните термични флуктуации, носителите на отрицателен заряд (*електрони*) и положителните (*дупки*) ще започнат да се движат съответно към положителния и отрицателния електроди. Ако полето е достатъчно силно, електроните и дупките ще се ускорят, и ще започнат да освобождават други свързани електрони, които също ще се ускоряват и така процесът става лавинен. Токът във веригата започва да нараства.

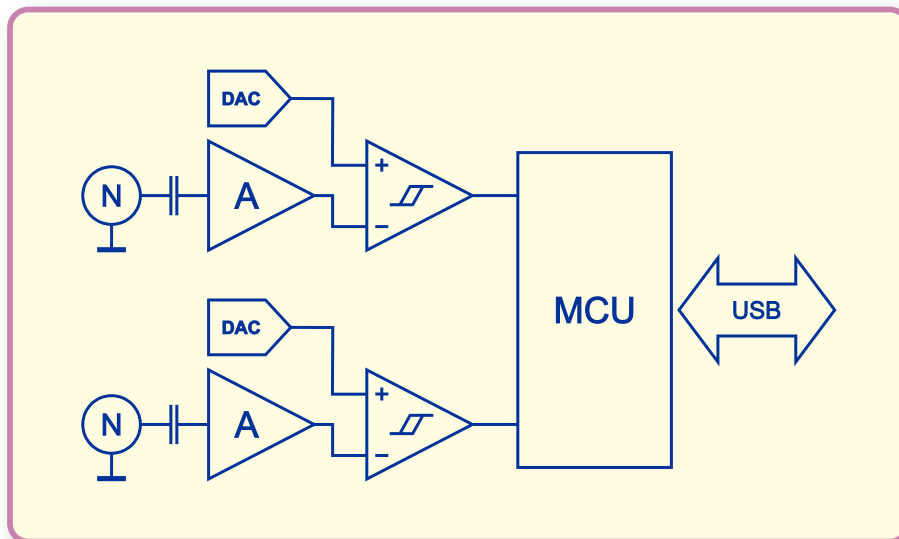
Генерираният шум в следствие на описания процес има сложна структура. Състои се предимно от квантов статичен шум (*shot noise*) плюс лавинния шум (*avalanche noise*), породен от вътрешното усилване. Обратното напрежение, при което възниква този процес, се нарича пробивно напрежение (*breakdown voltage*).



Волт-амперна характеристика на р-п преход.

2.3 Принципна схема

Принципната схема на устройството е показана по-долу.



GTR-01 принципна схема.

GTR-01 има два независими аналогови канала. При всеки от тях, сигналът от източника на шум се усилва до ниво, достатъчно за стабилно измерване и се свързва към единия от входовете на високоскоростен аналогов компаратор. Другият вход на компаратора се захранва от 12-битов цифрово-аналогов преобразувател и се използва за прецизно калибриране на канала. Изходите от аналоговите компаратори се свързват към микроконтролера и потока от нули и единици се обработва допълнително от процесора. Комуникацията на GTR-01 с компютъра се осъществява чрез стандартен USB интерфейс.

2.4 Технически спецификации

Спецификации	
Източник на ентропия	Лавинен шум в обратно свързан p-n преход
Брой независими канали	2
XOR	Да
Макс. скорост на трансфер	2.2 Mb/s
Работен температурен интервал	0 ÷ 50 °C
Платформа	Windows, Linux, Mac OS, Android
Размери	74 x 41 x 15 mm
Тегло	50 g

2.5 Инсталация на драйверите

За правилната си работа, GTR-01 се нуждае от инсталиране на USB драйвери. Можете да свалите драйверния пакет от секцията за техническа информация на страницата на продукта: <https://benevscitech.com/bg/quantum-random-generators-bg.html>

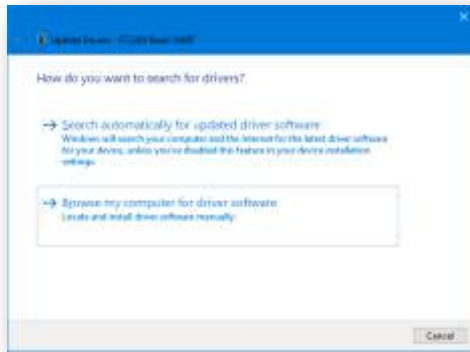
Следващият пример показва процедурата по инсталация на драйверния пакет на Windows 10. За останалите версии на Windows, процедурата следва подобни стъпки.



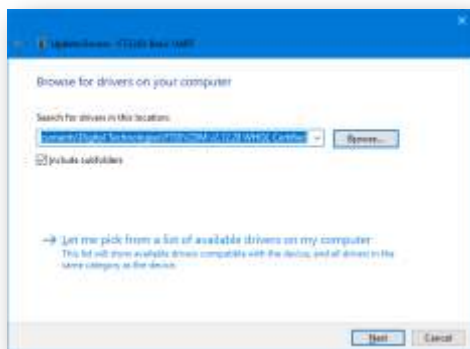
Стъпка 1. Свържете GTR-01 към компютъра. Когато устройството се свързва за първи път, Windows автоматично инсталира драйвер по подразбиране, но това не е правилния драйвер и затова в Device Manager устройството е маркирано с жълт знак.



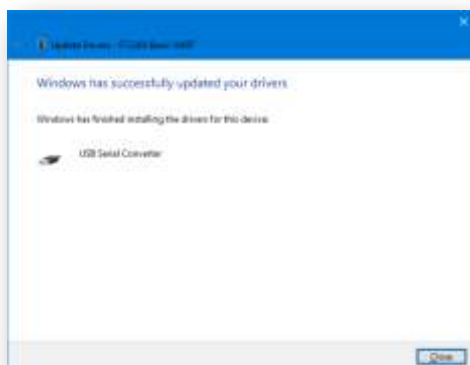
Стъпка 2. В Device Manager селектирайте устройството и натиснете десния бутон на мишката. От падащото меню изберете Properties. В страницата Driver кликнете върху бутона Update Driver.



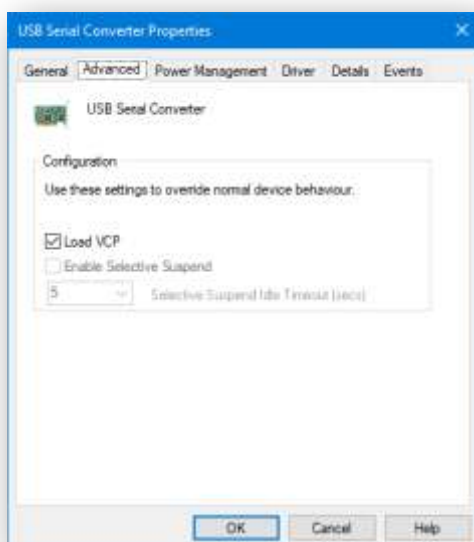
Стъпка 3. В следващия прозорец изберете Browse my computer for driver software.



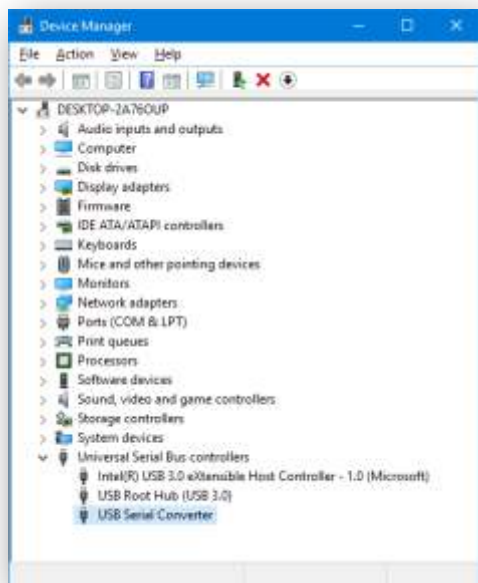
Стъпка 4. Натиснете бутона Browse и посочете папката с драйверите.



Стъпка 5. Изчакайте инсталацията да привърши.




Стъпка 6. По избор, след успешна инсталация на драйвера, в Device Manager щракнете с десния бутон върху устройството и изберете Properties от падащото меню. В раздела Advanced можете да изберете дали да заредите драйвера за виртуален COM порт и да комунирирате с GTR-01 като устройство за сериен COM порт.



След успешното инсталиране на драйверите, GTR-01 се вижда в Device Manager секцията USB controllers, като USB Serial Converter.

3. StudioGTR

StudioGTR е лесно за използване приложение с графичен интерфейс, базирано на MS Windows и е предназначено за наблюдение на работата, генериране на различни последователности от случайни числа и символи, статистическо тестване и калибриране на GTR-01. Оптимизираният потребителски интерфейс и опростената работа с програмата позволяват проверка на основната производителност на устройството и настройване на режимите за генериране. Можете да свалите приложението от секцията за техническа информация на страницата на продукта: <https://benevscitech.com/bg/quantum-random-generators-bg.html>

 **БЕЛЕЖКА:** За успешна комуникация между StudioGTR и GTR-01, USB драйверите трябва да са предварително инсталирани. За повече информация относно процедурата по инсталация на драйверния пакет вижте в предишната секция.


3.1 Системни изисквания

Изисквания	
Операционна система	Windows XP*, Vista, 7, 8, 8.1, 10, 11; 32/64-bit
Процесор	1 GHz или по-висока; 32/64-bit
Памет	1 GB или повече
Дисплей	1024 x 768 или по-добър
Свободно дисково пространство	50 MB или повече
USB	2.0, 3.0 или 3.1; един порт
Adobe Acrobat Reader	Версия 9.0 или по-нова

* Windows XP се нуждае от .NET Framework 2.0 или по-нова версия.

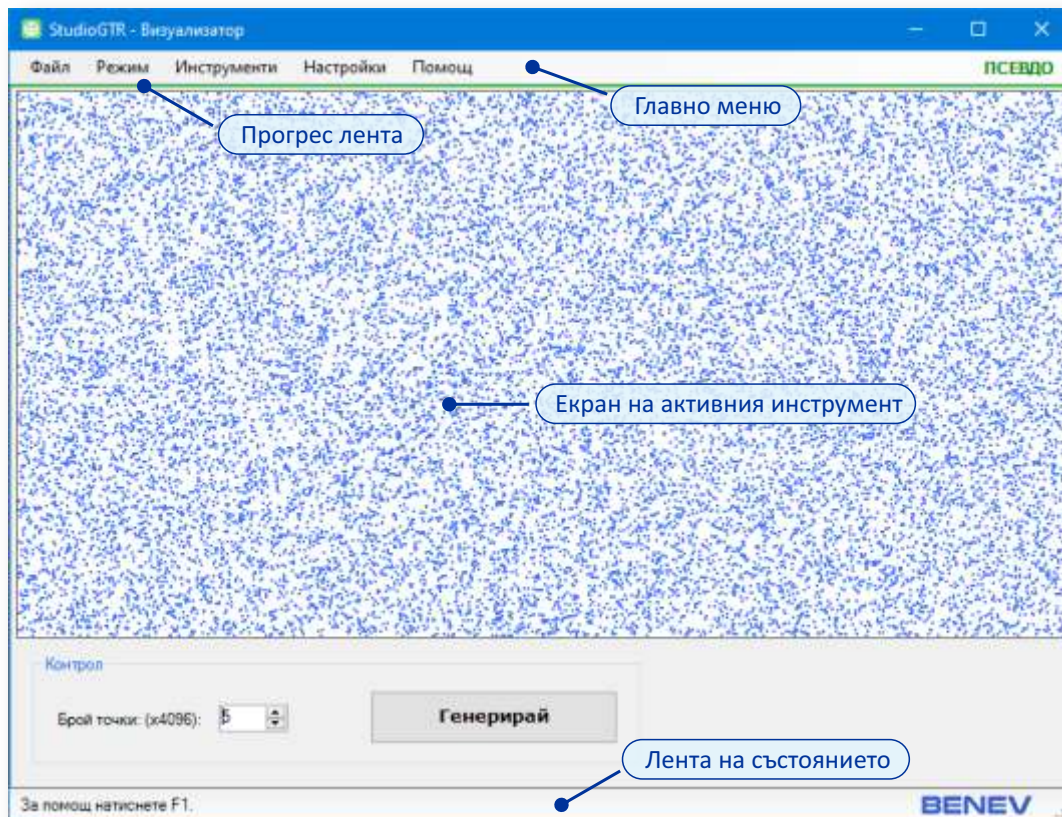
3.2 Инсталиране на програмата

StudioGTR не изисква специална инсталация. Разпакувайте файла StudioGTR.zip някъде във вашия компютър, например в C:\Program Files\StudioGTR и поставете пряк път към .exe файла на приложението на работния плот или менюто „Старт“. Това е всичко.

 **БЕЛЕЖКА:** Не променяйте структурата и съдържанието на разпакувания zip архив.

3.3 Главен екран

Главният екран на StudioGTR е показан на фигурата по-долу.




StudioGTR главен екран.

3.4 Общ преглед на възможностите на програмата

StudioGTR работи в два основни режима на генериране на случайни числа:

- **Псевдослучаен** – използва вградените в програмата софтуерни генератори на псевдослучайни числа. За да активирате този режим, от Главното меню изберете **Режим → Псевдослучаен**.
- **Истински** – използва GTR-01 като генератор на случайни числа. За да активирате този режим, от Главното меню изберете **Режим → Истински**.

 **БЕЛЕЖКА:** В дясната част на меню-лентата е показан текущият режим на работа и статуса на устройството.

StudioGTR предлага следните инструменти за работа със случайни числа и процеси:

- *Визуализатор* – предназначен за бърз визуален преглед на качеството на генерираните числа.
- *Цели числа* – генерира случайни последователности от цели числа в даден интервал.
- *Реални числа* – генерира случайни последователности от реални числа в даден интервал.
- *Символни низове* – генерира случайни последователности от символни низове.
- *Случаен шум* – генерира различни видове случаен шум.
- *Изображения* – генерира изображения със случайно разпределение на стойностите за отделните пиксели.
- *Лотарийни числа* – генерира печелифши числа за лотарийни билети и фишове.
- *Регистратор случайни събития* – генерира случайни числа през определен интервал от време и ги записва на твърдия диск на компютъра в непрекъснат автоматичен режим.
- *Статистически тестове* – прави автоматично статистическо тестване на използвания случаен генератор.
- *Калибриране* – калибрира GTR-01 в автоматичен или ръчен режим.



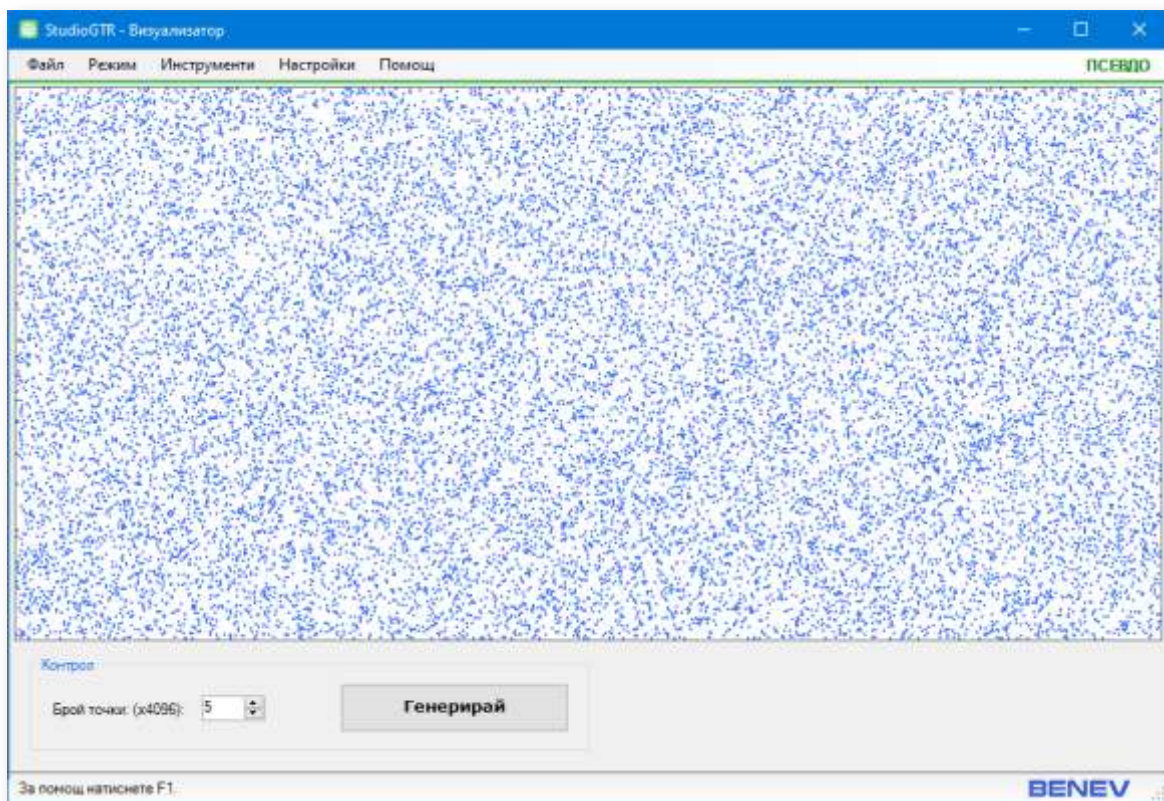
БЕЛЕЖКА: Можете да запишете изображение на екрана на програмата използвайки от Главното меню *Файл* → *Запис екран...* или CTRL+G.

3.5 Инструменти

3.5.1 Визуализатор



За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Визуализатор*. Позволява бърз визуален преглед на генерираните случайни числа.



StudioGTR Визуализатор.

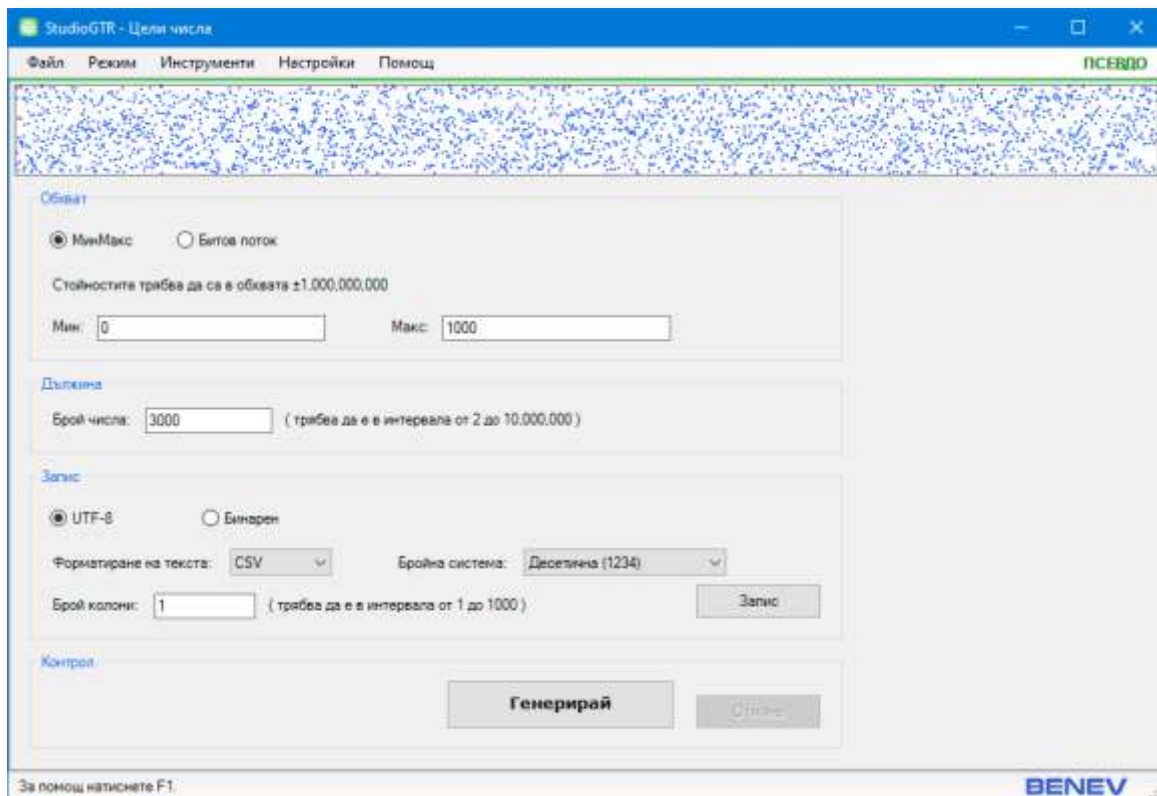


В секцията *Контрол* можете зададете броя на генерираните случайни числа.

3.5.2 Цели числа



За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Цели числа*. Позволява генериране на случайни последователности от цели числа с определена големина.



StudioGTR Цели числа.



В секцията *Обхват* можете да зададете обхвата на числата. Ако е избрана опцията *Битов поток*, числата са в обхват от -2^{31} до $+2^{31}-1$. Използвайте тази опция, за да генерирате битови последователности.



В секцията *Дължина* можете да посочите броя на генерираните числа.



В секцията *Запис* можете да запишете във файл генерираната последователност, като изберете вида на файла (текстови или бинарен). В случай на текстови запис, можете допълнително да посочите форматирането на данните (вида на разделителя), бройната система (десетична или шестнадесетична) и броя на колоните.

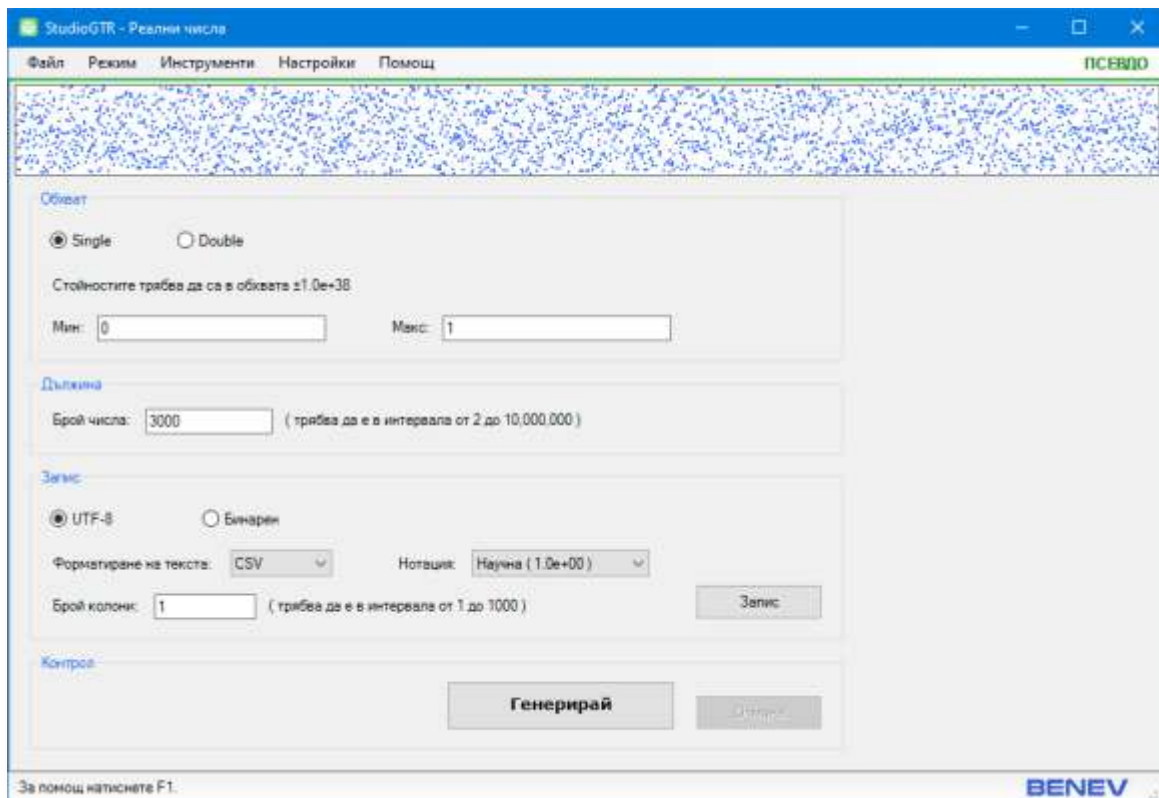


БЕЛЕЖКА: Генерираните цели числа са от тип *32-bit signed integer*.

3.5.3 Реални числа



За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Реални числа*. Позволява генериране на случайни последователности от числа с плаваща точка с определена големина.



StudioGTR Реални числа.



В секцията *Обхват* можете да зададете използваната точност (*Single 32-bit* или *Double 64-bit*) и обхвата на числата.



В секцията *Дължина* можете да посочите броя на генерираните числа.



В секцията *Запис* можете да запишете във файл генерираната последователност, като изберете вида на файла (текстови или бинарен). В случай на текстови запис, можете допълнително да посочите форматирането на данните (вида на разделителя), нотацията на записване (1.0e+00 или 1.0E+00) и броя на колоните.

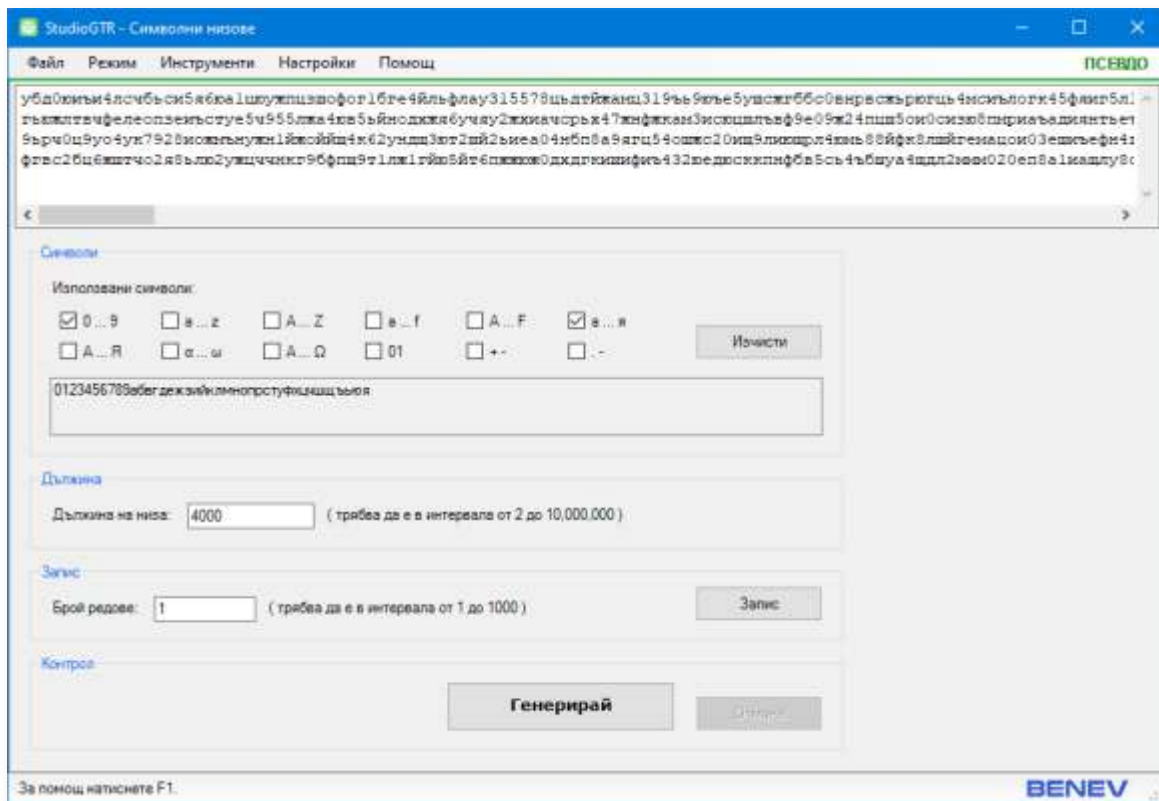


БЕЛЕЖКА: Генерираните числа използват вградените в *.NET* формати за числа с плаваща точка, съгласно съответните международни стандарти.

3.5.4 Символни низове



За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Символни низове*. Позволява генериране на случайни последователности от символни низове.



StudioGTR Символни низове.



В секцията *Символи* можете да изберете комбинациите от групи символи, използвани при генерирането на низа. Възможните символи са:

- 0123456789
- abcdefghijklmnopqrstuvwxyz
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- abcdef
- ABCDEF
- абвгдежзийклмнопрстуфхцчшщъьюя
- АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЮЯ
- αβγδεζηθικλμνξοπρστυφχψω
- ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
- 01
- +-
- .-



В секцията *Дължина* можете да посочите дължината на низа.




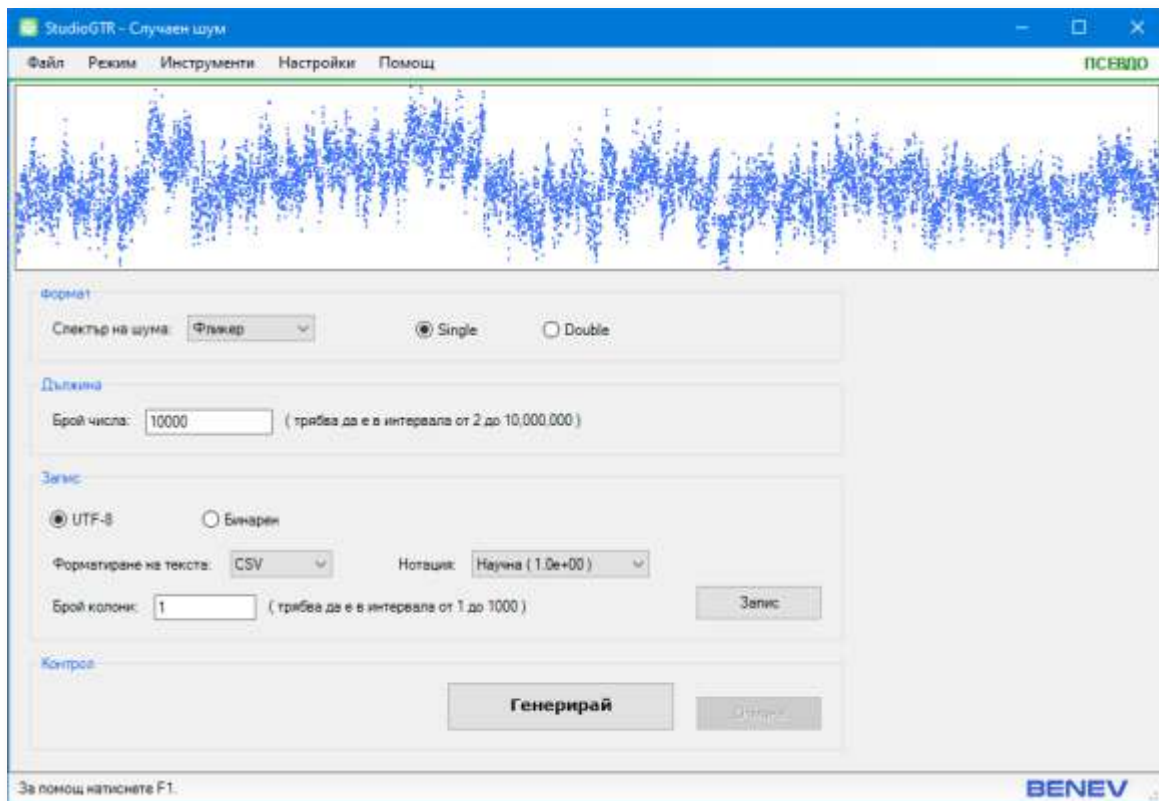
В секцията *Запис* можете да изберете броя на редовете.




БЕЛЕЖКА: Записът е винаги в текстови формат.

3.5.5 Случаен шум


 За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Случаен шум*. Позволява генериране на няколко вида случаен шум.




StudioGTR Случаен шум.

 В секцията *Формат* можете да посочите вида на генерирания шум и формата на числата (Single 32-bit или Double 64-bit). StudioGTR генерира следните видове случаен шум:

- **Бял шум.** Програмата генерира $N(0, 1)$ чрез трансформация на *Box-Muller*.
- **Фликер (Flicker noise).** Известен и като *Розов шум* или $1/f$ шум. StudioGTR използва алгоритъма показан в Gardner M. „Mathematical games-white and brown music, fractal curves and one-over-f fluctuations“, Scientific American, 238(4):16–32, 1978.
- **Браунов (Червен) шум.** Известен и като *Random Walk*.

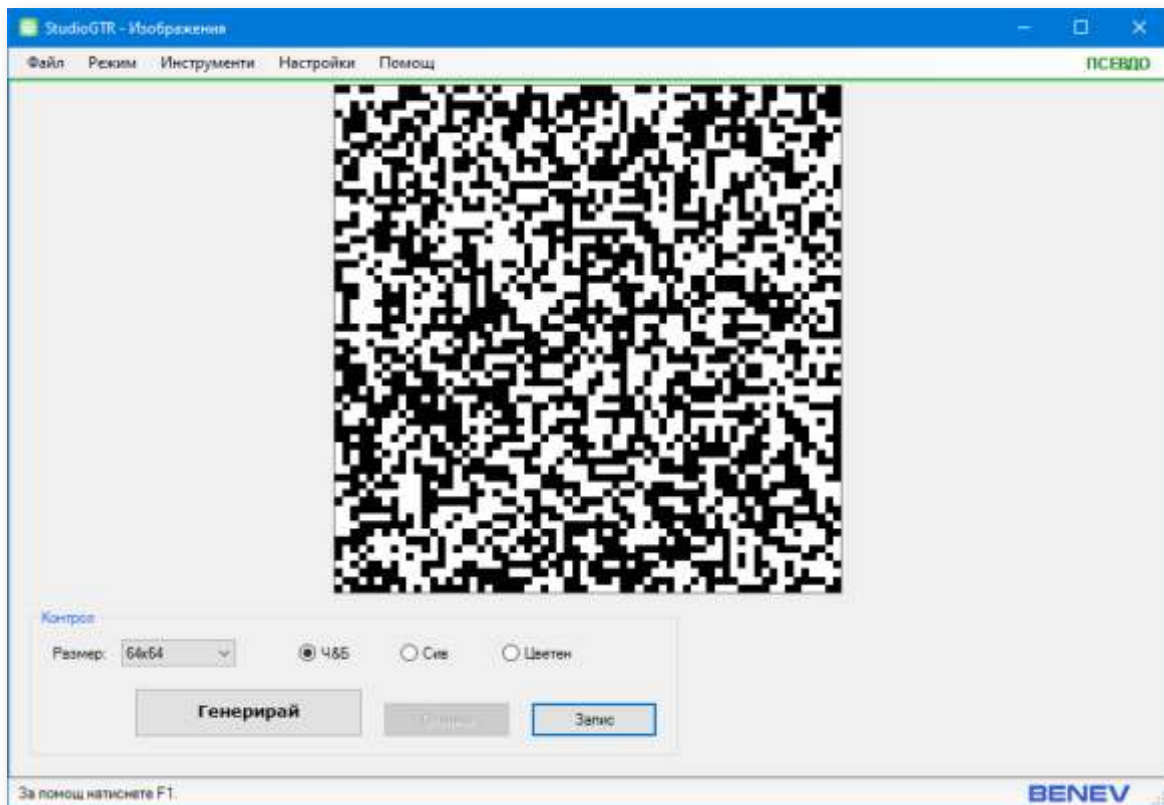
 В секцията *Дължина* можете да посочите броя на генерираните числа.

 В секцията *Запис* можете да запишете във файл генерирания шум, като изберете вида на файла (текстови или бинарен). В случай на текстови запис, можете допълнително да посочите форматирането на данните (вида на разделителя), нотацията на записване (1.0e+00 или 1.0E+00) и броя на колоните.

3.5.6 Изображения



За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Изображения*. Генерира квадратни изображения със случайно разпределение на стойностите на отделните пиксели.




StudioGTR Изображения.

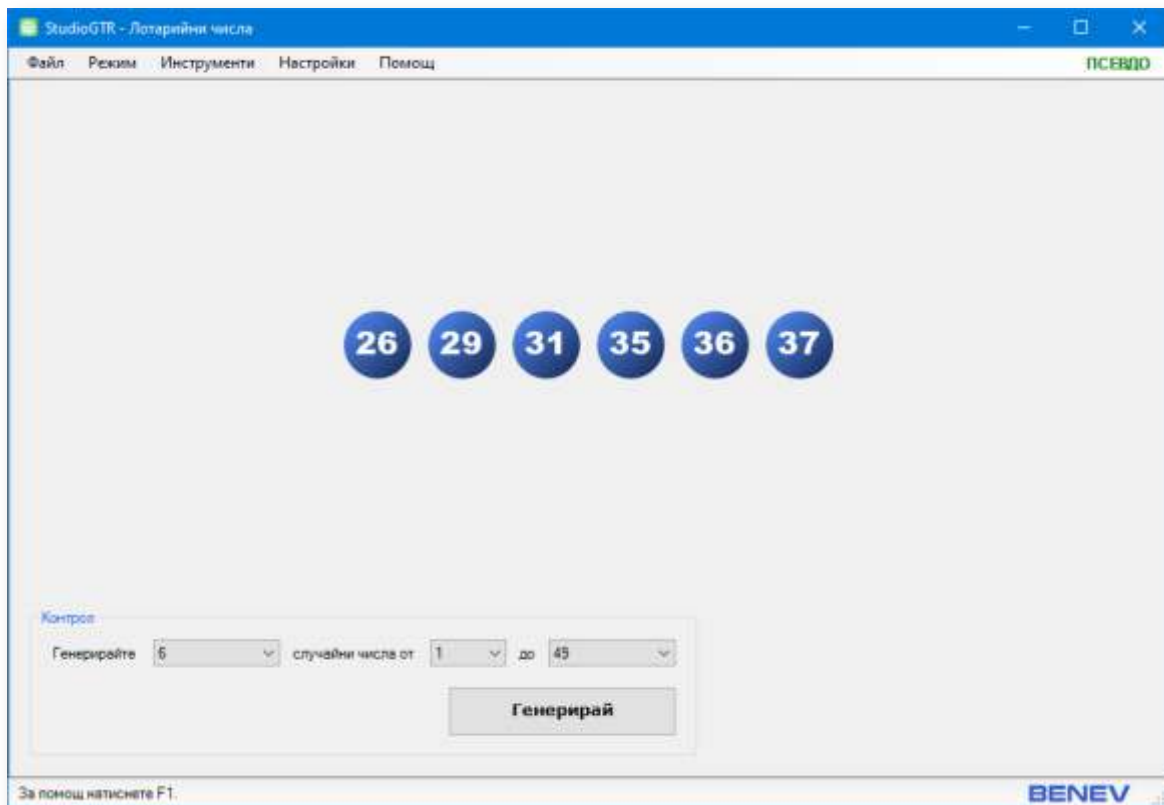


В секцията *Контрол* можете да зададете размера на изображението и вида му:


- **Ч&Б.** Пикселите приемат само стойностите 0 (черно) или 255 (бяло).
- **Сив.** Пикселите могат да приемат монохромни стойности между 0 (черно) и 255 (бяло).
- **Цветен.** Пикселите могат да приемат цветни RGB стойности, като всяка от трите компоненти на цвета (червена, зелена и синя) може да се изменя между 0 и 255.

3.5.7 Лотарийни числа

 За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Лотарийни числа*. Генерира печеливши числа за лотарийни билети и фишове.



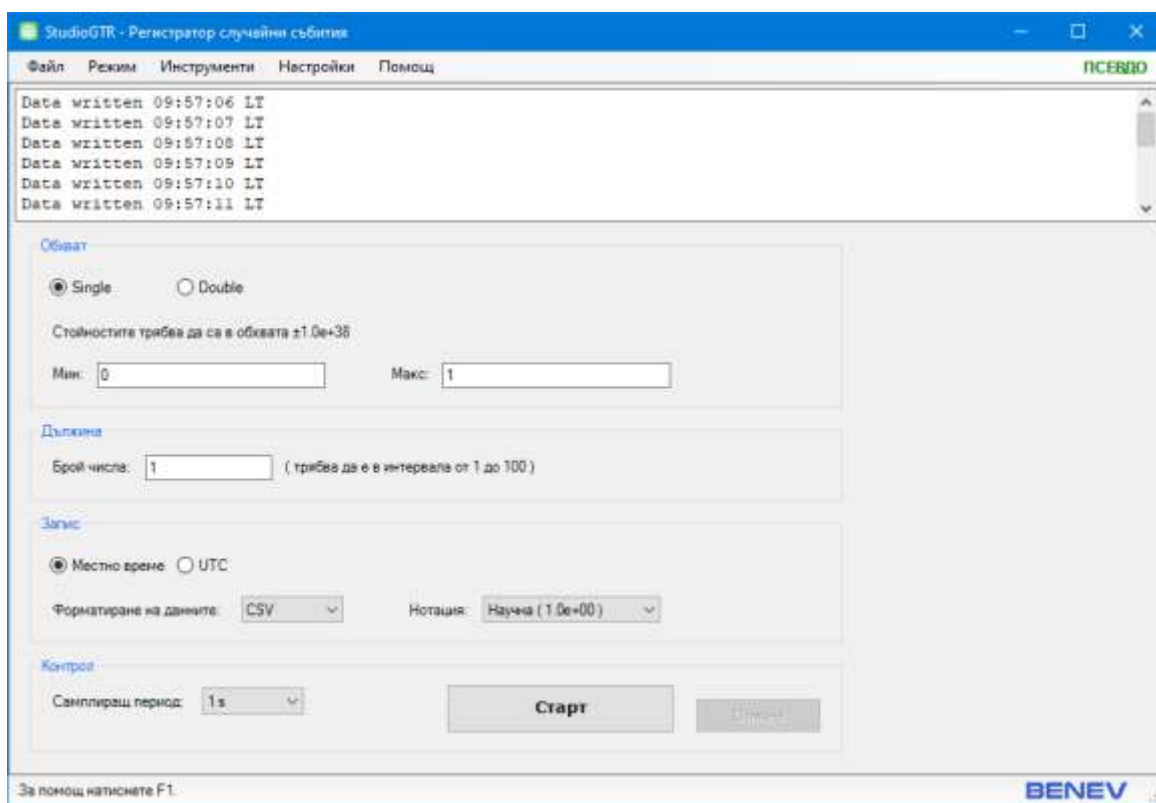
StudioGTR Лотарийни числа.

 В секцията *Контрол* можете да зададете броя на генерираните числа, началната им стойност (0 или 1) и диапазона от стойности, измежду който да се генерират. Например, за играта Тото 6 от 49 изберете шест числа със стойности от 1 до 49, за играта Тото 5 от 35 изберете 5 числа със стойности от 1 до 35 и т.н.

3.5.8 Регистратор случайни събития



За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Регистратор случайни събития*. Позволява генериране на случайни числа през определен интервал от време, в непрекъснат режим и автоматичния им запис на твърдия диск на компютъра. Инструментът има смисъл само в режим *Истински* и е подходящ за наблюдение влиянието на природни феномени върху работата на GTR-01. В същият дух на мисли, позволява използването на GTR-01 за регистрация на такива феномени.



StudioGTR Регистратор случайни събития.



В секцията *Обхват* можете да зададете използваната точност (*Single* – 32-bit или *Double* – 64-bit) и обхвата на числата.



В секцията *Дължина* можете да посочите броя на генерираните числа.



В секцията *Запис* можете да посочите вида на отчитаното време (*Локално* или *Universal Time Coordinated, UTC*), форматирането на данните (вида на разделителя) и нотацията на записване (1.0e+00 или 1.0E+00).



БЕЛЕЖКА: Записът е винаги в текстови формат. За избор на папка за запис, вижте секция 3.8.

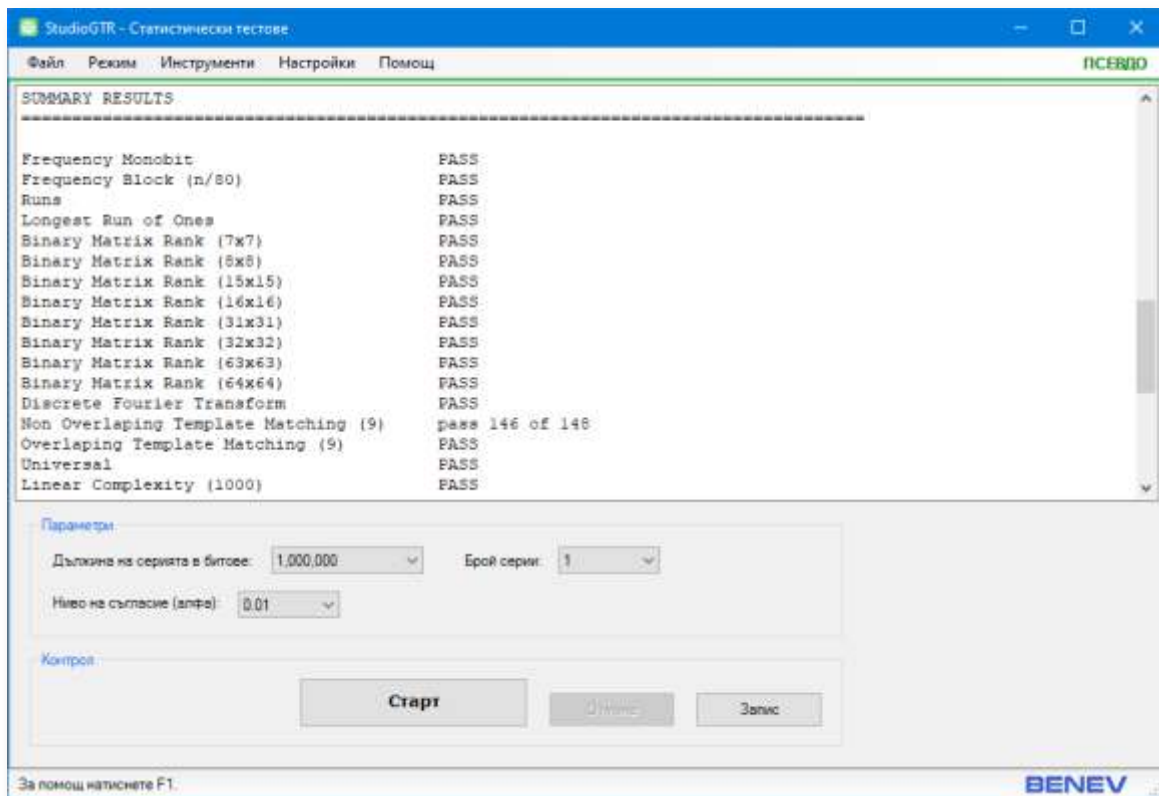


В секцията *Контрол* можете да зададете периода на семплиране между 1 s и 60 минути.

3.5.9 Статистически тестове



За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Статистически тестове*. Позволява провеждането на статистически тестове върху текущия генератор зададен в програмата. Използва специално разработена батерия от статистически тестове.



StudioGTR Статистически тестове.



В секцията *Параметри* можете да зададете дължината на серията в битове, броя серии използвани в тестовата процедура и статистическото ниво на съгласие (алфа).




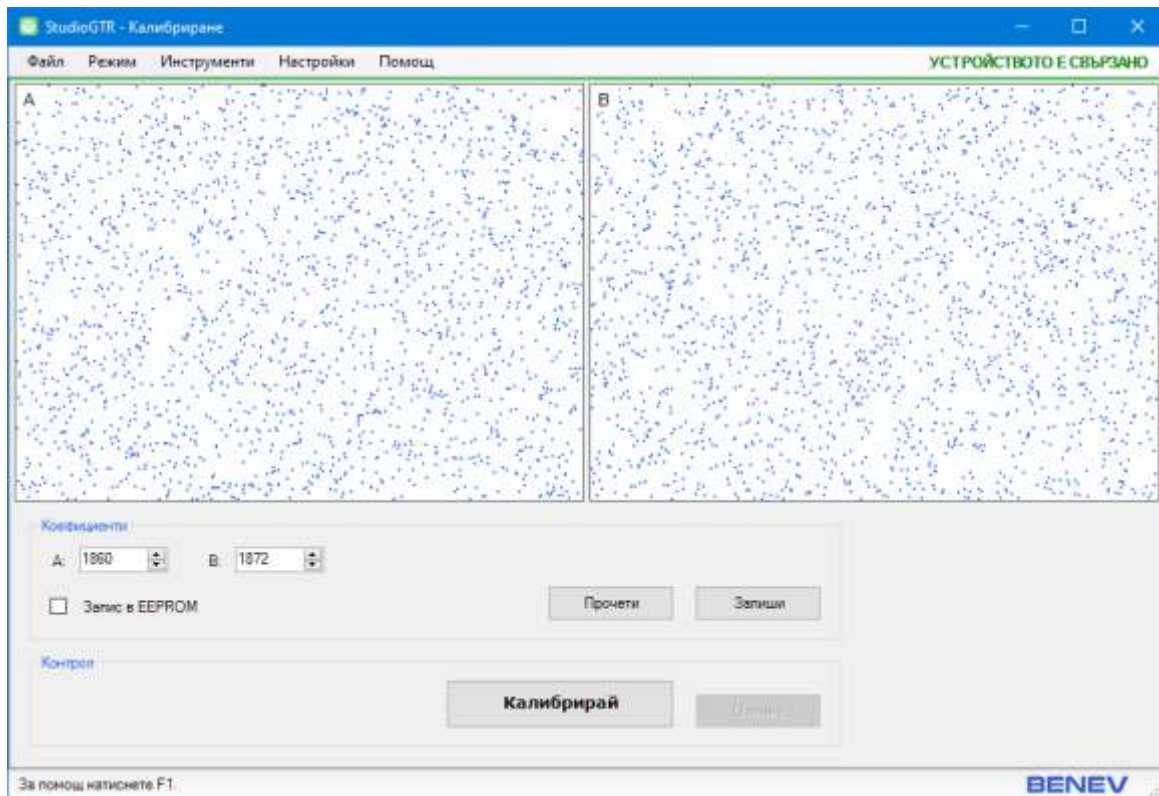
БЕЛЕЖКА: При голяма дължина на серията в битове и голям брой серии, тестването може да отнеме десетки часове.




БЕЛЕЖКА: StudioGTR използва батерията от статистически тестове предложена от *National Institute of Standards and Technology, NIST*. За повече информация вижте “*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*”, <https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final>


3.5.10 Калибриране


 За да активирате този инструмент, от Главното меню изберете *Инструменти* → *Калибриране*. Позволява калибрирането на GTR-01 в ръчен или автоматичен режим.




StudioGTR Калибриране.

 В секцията *Коефициенти* можете да прочетете текущите стойности за калибровъчните коефициенти на двата хардуерни канала на генератора, да зададете нови стойности или да ги запишете в устройството. При отменното поле *Запис в EEPROM*, коефициентите се записват в енергонезависимата памет на устройството.

 От секцията *Контрол* можете да стартирате автоматична процедура на калибриране и след това, ако желаете, да запишете получените стойности на калибровъчните коефициенти в паметта на устройството.

 **БЕЛЕЖКА:** Инструментът е активен само в режим *Истински*.

 **БЕЛЕЖКА:** GTR-01 не имплементира функция *Factory Reset*.

3.6 Настройки

3.6.1 Канал



За да зададете активния хардуерен канал на GTR-01, от Главното меню изберете *Настройки → Канал*. Програмата позволява следните опции:

- *Канал А*. Данните се генерират само от канал А.
- *Канал В*. Данните се генерират само от канал В.
- *XOR (Канал А XOR Канал В)*. Логическо XOR между битовете на двата канала (режим по подразбиране).
- *XOR плюс*. Както при горния режим, но с някои подобрения в качеството, за сметка на леко намаляване на скоростта на генериране.
- *XOR DeBias*. Както при режима по подразбиране, но с приложен алгоритъм на *John von Neumann* за изравняване (*debiasing*). Скоростта на генериране при този режим е занижена.

3.6.2 Трансфер



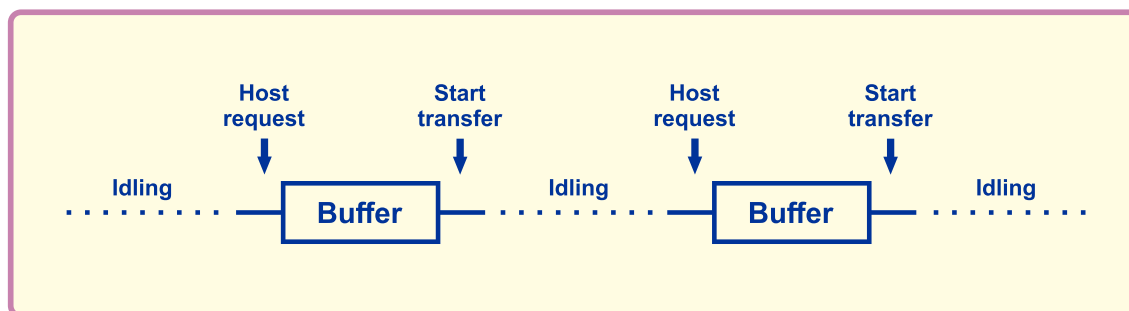
За да зададете вида на трансфера на данни от GTR-01, от Главното меню изберете *Настройки → Трансфер*. Програмата позволява следните опции:

- *Синхронен* (по подразбиране).
- *Асинхронен*.



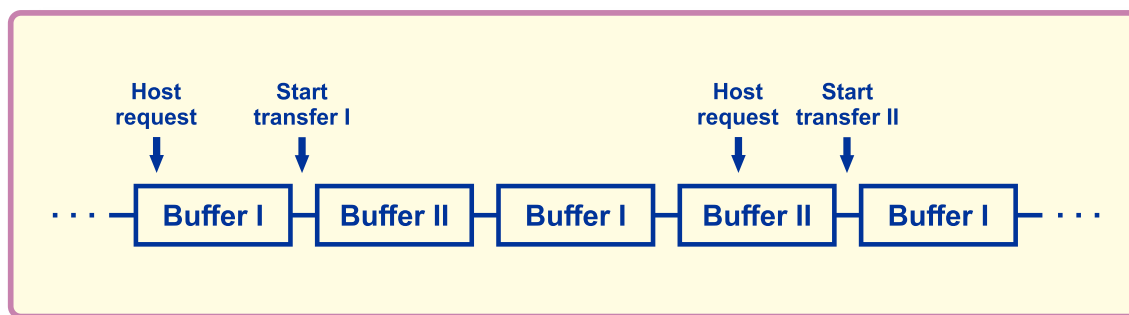
БЕЛЕЖКА: Вътрешно, GTR-01 имплементира два основни режима на генериране на данни.

В така наречения *Синхронен режим* данните се събират в единичен буфер всеки път, когато е направена заявка за данни от страна на компютъра. По този начин устройството (GTR-01) работи на празен ход, докато не бъде получена нова заявка и пакетите данни се синхронизират с работата на хоста, осигурявайки добра предвидимост и прецизност във времето. Този режим е подходящ в ситуации, когато точните моменти на събиране на данни трябва да се контролират от компютъра.



Синхронен режим на работа.

От друга страна, така нареченият *Асинхронен режим* използва схема за събиране на данните наречена *пинг-понг*. Данните се натрупват непрекъснато в два отделни буфера в паметта и независимо от заявките на хоста. След като единият буфер бъде запълнен, събирането продължава в другия без прекъсване и т.н. Двата буфера се сменят постоянно, един след друг, откъдето идва и името *пинг-понг*. Когато пристигне заявката от страна на компютъра, прехвърлянето на последния готов буфер, запълнен със свежи данни, може да започне незабавно и по този начин не се губи време в чакане буферът да бъде тепърва запълнен. Този режим е подходящ в случаите, когато основната цел е максимална скорост на работа с възможно най-висока скорост на трансфер.



Асинхронен режим на работа.

3.6.3 Псевдослучаен генератор



👉 За да зададете типа на псевдослучайния генератор използван от StudioGTR, от Главното меню изберете *Настройки → Псевдослучаен генератор*. Програмата позволява следните опции:

- *.NET* (по подразбиране). Използва стандартния генератор на *.NET*
- *.NET Crypto*. Използва стандартния криптографски сигурен генератор на *.NET*
- *LCG (Linear congruential generator)*.
- *MWC (Multiply-with-carry)*. Генератор изобретен от *George Marsaglia*.
- *Xorshift*. Генератор изобретен от *George Marsaglia*.
- *Combo (LCG+MWC+Xorshift)*.

3.7 Многоезичност

StudioGTR е програма разработена с вградена многоезична поддръжка.



За да смените текущия език на програмата:

- От Главното меню отворете секцията [Настройки](#) → [Език](#) и посочете избирания от Вас език.
- В появяващия се диалогов прозорец потвърдете избора си, след което StudioGTR автоматично ще се затвори и стартира наново с обновените езикови настройки.



БЕЛЕЖКА: Когато програмата се стартира за първи път на компютъра, активният език по подразбиране е *English*.



БЕЛЕЖКА: Помощният файл на програмата се стартира във версията с текущия език на приложението.

3.8 Предпочитания за програмата

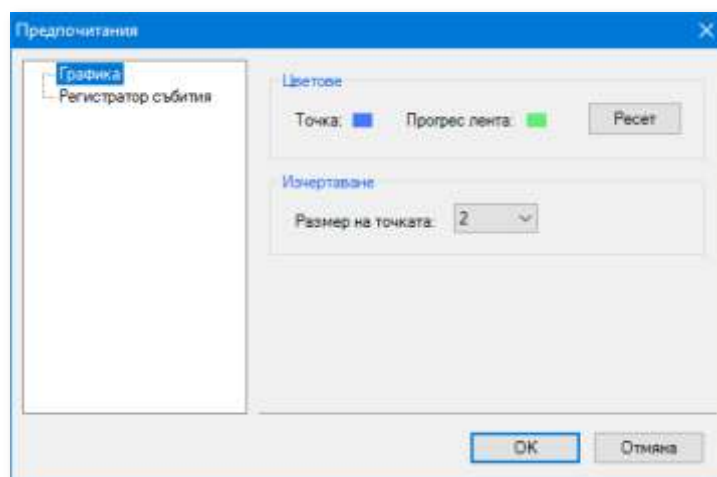
StudioGTR предоставя възможност за задаване на ползвателски предпочитанията за някои от работните параметри на програмата.



За да стартирате диалоговия прозорец с предпочитанията, от Главното меню изберете *Настройки → Предпочитания...*

- В страницата *Графика* изберете цвета за точките и прогрес лентата, както и размера на точката. Кликнете върху *Ресет* бутона, за да зададете цветовете по подразбиране.
- В страницата *Регистратор събития* посочете директорията за автоматичен запис на данните.

Когато сте готови натиснете ОК бутона. Настройките се променят незабавно. При следващото си стартиране StudioGTR ще се зареди с новите предпочитания.





Диалогов прозорец с предпочитания.

4. API на ниско ниво

GTR-01 е разработен като хардуерен източник на истински случайни числа. Може да се използва в области, изискващи високо качество на генерираните данни. И ако приложението StudioGTR позволява да се извършат някои основни наблюдения и процедури за калибриране, то пълният набор от всички възможности, вградени в GTR-01, може да бъде използван само чрез комуникиране с устройството на ниско ниво. Разбира се, този подход изисква някои основни умения в програмирането. Вижте Приложение А и Приложение В към това ръководство за повече примери с използване на Microsoft C#.


GTR-01 използва FTDI USB интерфейсен чип. Моля, прочетете документацията за драйверите и бележките за инсталирането им на сайта на производителя на драйверите.

 **БЕЛЕЖКА:** Комуникацията с GTR-01 изисква USB драйверите да бъдат инсталирани. За повече информация как да инсталирате пакета с драйвери, моля, вижте раздел 3.5.

 **БЕЛЕЖКА:** За максимална скорост на трансфер използвайте асинхронните IO команди. В останалите случаи използвайте синхронен трансфер.

4.1 GTR-01 IO команди

Описанието на входно-изходните команди на ниско ниво за GTR-01 IO следва по-долу.

 **ВАЖНО:** Винаги трябва да записвате пакет от 64 байта в GTR-01 и да четете обратно пакет от 4096 байта!

IO Команди	Стойност
GET_RND_CHANNEL_A	0x01
GET_RND_CHANNEL_A_CALIBRATION	0x02
GET_RND_CHANNEL_B	0x11
GET_RND_CHANNEL_B_CALIBRATION	0x12
GET_RND_NORMAL	0x21
GET_RND_NORMAL_PLUS	0x22
GET_RND_NORMAL_DEBIAS	0x23
GET_RND_CHANNEL_A_ASYNC	0x31
GET_RND_CHANNEL_A_CALIBRATION_ASYNC	0x32
GET_RND_CHANNEL_B_ASYNC	0x41
GET_RND_CHANNEL_B_CALIBRATION_ASYNC	0x42
GET_RND_NORMAL_ASYNC	0x51
GET_RND_NORMAL_PLUS_ASYNC	0x52
GET_RND_NORMAL_DEBIAS_ASYNC	0x53
SET_CALIBRATION	0x80
GET_CALIBRATION	0x81
GET_DEVICE_ID	0x82
EEPROM_WRITE*	0x55

* Командата трябва да бъде записана във втория байт от пакета. Вижте обяснението по-долу.

4.1.1 GET_RND_CHANNEL_A

Запис:

Byte0	0x01
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А, синхронен трансфер
-------------------	-------------------------------------------------

4.1.2 GET_RND_CHANNEL_A_CALIBRATION

Запис:

Byte0	0x02
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А, синхронен трансфер. Байтовете са генерирани „както са“ без никакви софтуерни подобрения. Използвайте тази команда при калибриране на канал А.
-------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.3 GET_RND_CHANNEL_B

Запис:

Byte0	0x11
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал В, синхронен трансфер.
-------------------	--------------------------------------------------

4.1.4 GET_RND_CHANNEL_B_CALIBRATION

Запис:

Byte0	0x12
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал В, синхронен трансфер. Байтовете са генерирани „както са“ без никакви софтуерни подобрения. Използвайте тази команда при калибриране на канал В.
-------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.5 GET_RND_NORMAL

Запис:

Byte0	0x21
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А XOR канал В, синхронен трансфер.
-------------------	--------------------------------------------------------------

4.1.6 GET_RND_NORMAL_PLUS

Запис:

Byte0	0x22
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А XOR канал В, синхронен трансфер. Идентична на GET_RND_NORMAL с някои софтуерни подобрения в качеството на изходните данни.
-------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.7 GET_RND_NORMAL_DEBIAS

Запис:

Byte0	0x23
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А XOR канал В, синхронен трансфер. Използва алгоритъм на John von Neumann за отстраняване на отклонението (bias).
-------------------	---------------------------------------------------------------------------------------------------------------------------------------------

4.1.8 GET_RND_CHANNEL_A_ASYNC

Запис:

Byte0	0x31
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А, асинхронен трансфер.
-------------------	---------------------------------------------------

4.1.9 GET_RND_CHANNEL_A_CALIBRATION_ASYNC

Запис:

Byte0	0x32
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А, асинхронен трансфер. Байтовете са генерирани „както са“ без никакви софтуерни подобрения. Използвайте тази команда при калибриране на канал А.
-------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.10 GET_RND_CHANNEL_B_ASYNC

Запис:

Byte0	0x41
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал В, асинхронен трансфер.
-------------------	---------------------------------------------------

4.1.11 GET_RND_CHANNEL_B_CALIBRATION_ASYNC

Запис:

Byte0	0x42
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал В, асинхронен трансфер. Байтовете са генерирани „както са“ без никакви софтуерни подобрения. Използвайте тази команда при калибриране на канал В.
-------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.12 GET_RND_NORMAL_ASYNC

Запис:

Byte0	0x51
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А XOR канал В, асинхронен трансфер.
-------------------	---------------------------------------------------------------

4.1.13 GET_RND_NORMAL_PLUS_ASYNC

Запис:

Byte0	0x52
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А XOR канал В, асинхронен трансфер. Идентична на GET_RND_NORMAL_ASYNC с някои софтуерни подобрения в качеството на изходните данни.
-------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------

4.1.14 GET_RND_NORMAL_DEBIAS

Запис:

Byte0	0x53
Byte1 to Byte63	Без значение

Четене:

Byte0 to Byte4095	Случайни байтове от канал А XOR канал В, асинхронен трансфер. Използва алгоритъм на John von Neumann за отстраняване на отклонението (bias).
-------------------	----------------------------------------------------------------------------------------------------------------------------------------------

4.1.15 SET_CALIBRATION

Запис:

Byte0	0x80
Byte1	0x55 - Запис в EEPROM; Иначе - без запис в EEPROM
Byte2	Резервиран
Byte3	Резервиран
Byte4	Канал А калибр. стойност, 12-bit unsigned, byte0
Byte5	Канал А калибр. стойност, 12-bit unsigned, byte1
Byte6	Канал В калибр. стойност, 12-bit unsigned, byte0
Byte7	Канал В калибр. стойност, 12-bit unsigned, byte1
Byte8 to Byte63	Без значение

Четене:

Byte0	EEPROM запис статус, 0 означава ОК
Byte1 to Byte4095	Няма стойност

4.1.16 GET_CALIBRATION

Запис:

Byte0	0x81
Byte1 to Byte63	Без значение

Четене:

Byte0	Канал А калибр. стойност, 12-bit unsigned, byte0
Byte1	Канал А калибр. стойност, 12-bit unsigned, byte1
Byte2	Канал В калибр. стойност, 12-bit unsigned, byte0
Byte3	Канал В калибр. стойност, 12-bit unsigned, byte1
Byte4 to Byte4095	Няма стойност

4.1.17 GET_DEVICE_ID

Запис:

Byte0	0x82
Byte1 to Byte63	Без значение

Четене:

Byte0	Сериен ID, 64-bit unsigned integer, byte0
Byte1	Сериен ID, 64-bit unsigned integer, byte1
Byte2	Сериен ID, 64-bit unsigned integer, byte2
Byte3	Сериен ID, 64-bit unsigned integer, byte3
Byte4	Сериен ID, 64-bit unsigned integer, byte4
Byte5	Сериен ID, 64-bit unsigned integer, byte5
Byte6	Сериен ID, 64-bit unsigned integer, byte6
Byte7	Сериен ID, 64-bit unsigned integer, byte7
Byte8 to Byte4095	Няма стойност

Приложение А: Статистически тестове

Качеството на генерираните от GTR-01 случайни числа е валидирано с използването на следните батерии от статистически тестове :

Тест	Уебсайт
NIST STS	https://csrc.nist.gov/projects/random-bit-generation
TestU01	http://simul.iro.umontreal.ca/testu01/tu01.html
Diehard	https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/



БЕЛЕЖКА: За повече информация свързана с процедурата на тестване четете тук:
<https://benevscitech.com/bg/statistical-testing-random-generators-bg.html>

Приложение В: FTDI IO примерен код на C#

Това е примерен код с използване на C# и технологията P/Invoke за “обвиване” на някои основни IO методи, първоначално написани на C и описани в *D2XX Programmer's Guide* на <https://ftdichip.com> Кодът, показан по-долу, се намира във файла GTR_IO_wrap.cs, като част от архива StudioGTR.zip.

```
using System;
using System.Text;
using System.Runtime.InteropServices;

namespace GTR
{
    // FT_STATUS enumeration.
    public enum FT_STATUS : uint
    {
        FT_OK = 0,
        FT_INVALID_HANDLE,
        FT_DEVICE_NOT_FOUND,
        FT_DEVICE_NOT_OPENED,
        FT_IO_ERROR,
        FT_INSUFFICIENT_RESOURCES,
        FT_INVALID_PARAMETER,
        FT_INVALID_BAUD_RATE,
        FT_DEVICE_NOT_OPENED_FOR_ERASE,
        FT_DEVICE_NOT_OPENED_FOR_WRITE,
        FT_FAILED_TO_WRITE_DEVICE,
        FT_EEPROM_READ_FAILED,
        FT_EEPROM_WRITE_FAILED,
        FT_EEPROM_ERASE_FAILED,
        FT_EEPROM_NOT_PRESENT,
        FT_EEPROM_NOT_PROGRAMMED,
        FT_INVALID_ARGS,
        FT_NOT_SUPPORTED,
        FT_OTHER_ERROR
    }

    // FT_DEVICE_LIST_INFO_NODE struct.
    [StructLayout(LayoutKind.Sequential)]
    public struct FT_DEVICE_LIST_INFO_NODE
    {
        public uint Flags;
        public uint Type;
        public uint ID;
        public uint LocId;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
        public string SerialNumber;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 64)]
        public string Description;
        public IntPtr ftHandle;
    }
}
```

```
// FTDI class.
public class FTDI
{
    // Flags.
    public const uint FT_LIST_NUMBER_ONLY = 0x80000000u;
    public const uint FT_LIST_BY_INDEX = 0x40000000u;
    public const uint FT_LIST_ALL = 0x20000000u;

    public const uint FT_OPEN_BY_SERIAL_NUMBER = 1u;
    public const uint FT_OPEN_BY_DESCRIPTION = 2u;
    public const uint FT_OPEN_BY_LOCATION = 4u;

    // Word Lengths.
    public const byte FT_BITS_8 = (byte)8;
    public const byte FT_BITS_7 = (byte)7;

    // Stop Bits.
    public const byte FT_STOP_BITS_1 = (byte)0;
    public const byte FT_STOP_BITS_2 = (byte)2;

    // Parity.
    public const byte FT_PARITY_NONE = (byte)0;
    public const byte FT_PARITY_ODD = (byte)1;
    public const byte FT_PARITY_EVEN = (byte)2;
    public const byte FT_PARITY_MARK = (byte)3;
    public const byte FT_PARITY_SPACE = (byte)4;

    // Flow Control.
    public const ushort FT_FLOW_NONE = (ushort)0x0000;
    public const ushort FT_FLOW_RTS_CTS = (ushort)0x0100;
    public const ushort FT_FLOW_DTR_DSR = (ushort)0x0200;
    public const ushort FT_FLOW_XON_XOFF = (ushort)0x0400;

    // Purge rx and tx buffers.
    public const uint FT_PURGE_RX = 1;
    public const uint FT_PURGE_TX = 2;

    // Classic Interface Methods.
    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_CreateDeviceInfoList(ref uint lpdwNumDevs);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_GetDeviceInfoList(
        [In, Out] FT_DEVICE_LIST_INFO_NODE[] pDest,
        ref uint lpdwNumDevs);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_ListDevices(ref uint pvArg1,
        uint pvArg2,
        uint dwFlags);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_Open(int iDevice,
        ref IntPtr ftHandle);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_Close(IntPtr ftHandle);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_Read(IntPtr ftHandle,
        [In, Out] byte[] lpBuffer,
        uint dwBytesToRead,
        ref uint lpdwBytesReturned);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_Write(IntPtr ftHandle,
        [In, Out] byte[] lpBuffer,
        uint dwBytesToWrite,
        ref uint lpdwBytesWritten);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_SetBaudRate(IntPtr ftHandle,
        uint dwBaudRate);

    [DllImport("FTD2XX.dll")]
    public static extern FT_STATUS FT_SetDivisor(IntPtr ftHandle,
        ushort usDivisor);
```

```
[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_SetDataCharacteristics(IntPtr ftHandle,
                                                         byte uWordLength,
                                                         byte uStopBits,
                                                         byte uParity);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_SetTimeouts(IntPtr ftHandle,
                                              uint dwReadTimeout,
                                              uint dwWriteTimeout);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_SetFlowControl(IntPtr ftHandle,
                                                ushort usFlowControl,
                                                byte uXon,
                                                byte uXoff);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_GetDeviceInfo(IntPtr ftHandle,
                                                ref uint pftType,
                                                ref uint lpdwID,
                                                StringBuilder pcSerialNumber,
                                                StringBuilder pcDescription,
                                                int pvDummy);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_GetDriverVersion(IntPtr ftHandle,
                                                  ref uint lpdwDriverVersion);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_GetLibraryVersion(ref uint lpdwDLLVersion);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_Purge(IntPtr ftHandle,
                                       uint dwMask);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_ResetDevice(IntPtr ftHandle);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_ResetPort(IntPtr ftHandle);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_CyclePort(IntPtr ftHandle);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_Rescan();

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_Reload(ushort wVID,
                                       ushort wPID);

// Extended API Functions.
[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_SetLatencyTimer(IntPtr ftHandle,
                                                  byte ucTimer);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_GetLatencyTimer(IntPtr ftHandle,
                                                  ref byte pucTimer);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_SetBitMode(IntPtr ftHandle,
                                             byte ucMask,
                                             byte ucMode);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_GetBitMode(IntPtr ftHandle,
                                             ref byte pucMode);

[DllImport("FTD2XX.dll")]
public static extern FT_STATUS FT_SetUSBParameters(IntPtr ftHandle,
                                                  uint dwInTransferSize,
                                                  uint dwOutTransferSize);
}
```

Приложение С: GTR-01 IO пример

Това е примерен код, описващ основната входно-изходна комуникация с GTR-01. Кодът е написан на C# и се намира в класа GTR_IO във файла GTR_IO_ex.cs. Примерът използва интерфейлната обвивка от файла GTR_IO_wrap.cs. И двата изходни файла с примерен код са част от архива StudioGTR.zip.

```
using System;

namespace GTR
{
    // GTR_IO class.
    public class GTR_IO
    {
        // Constants.
        private const ushort VID = 0x0403;
        private const ushort PID = 0x6015; // FTDI's original PID and VID.

        private const uint BYTES_TO_WRITE = 64u;
        private const uint BYTES_TO_READ = 4096u;

        private const byte EEPROM_WRITE_CONST = (byte) 0x55;

        public const byte GET_RND_CHANNEL_A_REQUEST = (byte) 0x01;
        public const byte GET_RND_CHANNEL_A_CALIBRATION_REQUEST = (byte) 0x02;
        public const byte GET_RND_CHANNEL_B_REQUEST = (byte) 0x11;
        public const byte GET_RND_CHANNEL_B_CALIBRATION_REQUEST = (byte) 0x12;
        public const byte GET_RND_NORMAL_REQUEST = (byte) 0x21;
        public const byte GET_RND_NORMAL_PLUS_REQUEST = (byte) 0x22;
        public const byte GET_RND_NORMAL_DEBIAS_REQUEST = (byte) 0x23;
        public const byte GET_RND_CHANNEL_A_ASYNC_REQUEST = (byte) 0x31;
        public const byte GET_RND_CHANNEL_A_CALIBRATION_ASYNC_REQUEST = (byte) 0x32;
        public const byte GET_RND_CHANNEL_B_ASYNC_REQUEST = (byte) 0x41;
        public const byte GET_RND_CHANNEL_B_CALIBRATION_ASYNC_REQUEST = (byte) 0x42;
        public const byte GET_RND_NORMAL_ASYNC_REQUEST = (byte) 0x51;
        public const byte GET_RND_NORMAL_PLUS_ASYNC_REQUEST = (byte) 0x52;
        public const byte GET_RND_NORMAL_DEBIAS_ASYNC_REQUEST = (byte) 0x53;

        private const byte SET_CALIBRATION_REQUEST = (byte) 0x80;
        private const byte GET_CALIBRATION_REQUEST = (byte) 0x81;
        private const byte GET_DEVICE_ID_REQUEST = (byte) 0x82;

        // Constructor.
        public GTR_IO()
        {
        }
    }
}
```

```
// Public methods.
public void GTR_Basic_IO()
{
    IntPtr handle = IntPtr.Zero;

    try
    {
        FT_STATUS status = FT_STATUS.FT_OK;
        uint numDevs = 0u;

        // Check for GTR device.
        status = FTDI.FT_CreateDeviceInfoList(ref numDevs);

        FT_DEVICE_LIST_INFO_NODE[] vDevInfo =
            new FT_DEVICE_LIST_INFO_NODE[numDevs];
        status = FTDI.FT_GetDeviceInfoList(vDevInfo, ref numDevs);

        string vidpid = String.Format("{0:X4}{1:X4}", GTR_IO.VID, GTR_IO.PID);

        if (status == FT_STATUS.FT_OK &&
            String.Format("{0:X8}", vDevInfo[0].ID) == vidpid)
        {
            // GTR is connected.
        }
        else
        {
            // There is something wrong or device is not connected.
        }

        // Open GTR device.
        if (FTDI.FT_Open(0, ref handle) != FT_STATUS.FT_OK)
        {
            throw new Exception("Device not found!");
        }

        // Initialize GTR device.
        FTDI.FT_SetBaudRate(handle, 3000000u);
        FTDI.FT_SetDataCharacteristics( handle,
                                         FTDI.FT_BITS_8,
                                         FTDI.FT_STOP_BITS_1,
                                         FTDI.FT_PARITY_NONE);
        FTDI.FT_SetFlowControl(handle, FTDI.FT_FLOW_RTS_CTS, 0, 0);
        FTDI.FT_SetTimeouts(handle, 3000u, 1000u);
        FTDI.FT_Purge(handle, FTDI.FT_PURGE_RX | FTDI.FT_PURGE_TX);

        // IO communication.
        uint bytesToWrite = GTR_IO.BYTES_TO_WRITE;
        uint bytesToRead = GTR_IO.BYTES_TO_READ;
        uint bytesWritten = 0u;
        uint bytesReturned = 0u;
        byte[] bufferWrite = new byte[bytesToWrite];
        byte[] bufferRead = new byte[bytesToRead];

        bufferWrite[0] = GTR_IO.GET_RND_NORMAL_REQUEST;

        status = FTDI.FT_Write(handle,
                               bufferWrite,
                               bytesToWrite,
                               ref bytesWritten);
        status = FTDI.FT_Read(handle,
                              bufferRead,
                              bytesToRead,
                              ref bytesReturned);
        if (status != FT_STATUS.FT_OK)
        {
            throw new Exception("USB communication problem!");
        }
    }
}
```

```
        // Displaying the result.
        for (int i = 0; i < bufferRead.Length; i++)
        {
            Console.Write("{0:x2}", bufferRead[i]);
        }
        Console.WriteLine();
    }
    catch (Exception exc)
    {
        // If something goes wrong, show the explanation here.
        Console.WriteLine(exc.Message);
    }
    finally
    {
        // When the IO is completed, close GTR device.
        FTDI.FT_Close(handle);
    }
}
}
```

Азбучен указател

Асинхронен режим	23	Област на обедняване	3
Бял шум	16	Обратно напрежение	3
Браунов шум	16	Пробивно напрежение	3
Box-Muller трансформация	16	Ping-pong	23
George Marsaglia	23	P-област	3
Драйвери	5, 6, 7	P-n преход	3
Debiasing	22, 29, 31	Розов шум	16
EEPROM	1, 21	Random Walk	16
John von Neumann	22, 29, 31	RGB	15
Калибрационни стойности	21	Синхронен режим	22
Квантов шум	3	Статичен квантов шум	3
Компаратор	4	Фликер шум	16
Криптография	1, 23	Xorshift	23
Лавинен шум	1, 3	Цифрово аналогов преобразувател	4
Linear congruential generator	23	Червен шум	16
Multiply-with-carry	23		
NIST	20		
N-област	3		

