Random
Portable random number generator for RW 6.0+

Contents

About

This module was created for ABB robots before ABB created the Rand() function. It is compatible with RW6.0. It hosts several functions to create Random numbers. It is still preferred to the new built in generator, because it is easier to use, although later releases rely on the built in versions, because has support.

Source Paper

This Random Number Generator is based on the algorithm in a FORTRAN version published by George Marsaglia and Arif Zaman, Florida State University; ref.: see original comments below. At the fhw (Fachhochschule Wiesbaden, W.Germany), Dept. of Computer Science, we have written sources in further languages (C, Modula-2 Turbo-Pascal(3.0, 5.0), Basic and Ada) to get exactly the same test results compared with the original FORTRAN version. April 1989 Karl-L. Noell NOELL@DWIFH1.BITNET and Helmut Weber WEBER@DWIFH1.BITNET

This random number generator originally appeared in "Toward a Universal Random Number Generator" by George Marsaglia and Arif Zaman. Florida State University Report: FSU-SCRI-87-50 (1987) It was later modified by F. James and published in "A Review of Pseudo-random Number Generators" THIS IS THE BEST KNOWN RANDOM NUMBER GENERATOR AVAILABLE. (However, a newly discovered technique can yield a period of 10^600. But that is still in the development stage.) It passes ALL of the tests for random number generators and has a period of 2^144, is completely portable (gives bit identical results on all machines with at least 24-bit mantissas in the floating point representation).

The algorithm is a combination of a Fibonacci sequence (with lags of 97 and 33, and operation "subtraction plus one, modulo one") and an "arithmetic sequence" (using subtraction).

Use IJ = 1802 & KL = 9373 to test the random number generator. The subroutine RANMAR should be used to generate 20000 random numbers.bThen display the next six random numbers generated multiplied by 4096*4096 If the random number generator is working properly, the random numbers should be:

```
6533892.0  14220222.0  7275067.0

6172232.0  8354498.0   10633180.0
```

Converted to ABB RAPID code in 2019 by Nathan Rapp

Release Notes

1.0

This was the first instance of the Random Number generator. Created before ABB built-in random number generator. 2019

1.1

Added Persistent Boolean to check if data array has been initialized. Changed internal array to persistent, so that the user does not need to initialize it every time.

1.2

Added a RandArrayIndex, so that the Random number generator can randomly work its way through an array.

1.3

Removed RandArrayIndex. Created Documentation for Random Module. MMC Pallet for pendant users. Created Add-in.

Local Module Function

About

These functions should not be changed. They are used to generate the random numbers based on the whitepaper. Changing of these functions may break the functionality of the module. Local module functions are only descriptions because the user shouldn't use or save change them.

RandomInitialize

Random Initialize routine for the random number generator. The Random number sequences created by these two seeds are of sufficient length to complete an entire calculation with. For example, if several difference groups are working on different parts of the same calculation, each group could be assigned its own IJ seed. This would leave each group with 30000 choices for the second seed. That is to say, this random number generator can create 900 million different subsequences – with each subsequence having length of approximately 10^30.

Log

Returns the Natural Logarithm of a number. Used in the RandomGausian function.

Global Functions

About

These are functions intended for users. Then are global. The have full examples & descritptions.

RandomFloat

Usage

RandomFloat is used to return a random decimal number bounded between a lower value and an upper value.

Basic Example

```
PROC Example()
  !Examples of Random Floats , 5 to 10:
  FOR i FROM 1 TO 10 DO
    ErrWrite\i,"Float: "+ValToStr(RandomFloat(5,10)),"";
  ENDFOR
 ENDPROC
```

Return Value

Data Type: Num

The Random decimal number between the lower & upper.

Arguments

Lower

> Lower Range Limit
>
> DataType: Num
>
> The lower number in the Range .

Upper

> Upper Range Limit
>
> DataType: Num
>
> The Upper number in the Range .

Syntax

RandomFloat '('

> [Lower ':='] <expression (**IN**) of num>
>
> [Upper ':='] <expression (**IN**) of num> ')'

RandomGausian

RandomGausian is used to return a random decimal normally distributed around a mean, given a standard deviation.

Basic Example

```
PROC Example()
  !Examples of Random numbers, Normally distributed
  !around the mean of 10, with a Std. Deviation of 0.3;
  FOR i FROM 1 TO 10 DO
   ErrWrite\i,"Gaus: "+ValToStr(RandomGausian(10,0.3)),"";
  ENDFOR
 ENDPROC
```

Return Value

Data Type: Num

The Random decimal number normally distributed about the mean with a standard deviation

Arguments

Mean

Mean also known as Average

DataType: Num

The average or most common number in a collection of numbers

Stddev

Standard Deviation

DataType: Num

The amount of variance or dispersion of a set of values

Syntax

RandomGausian ‘(’

[Mean ‘:=’] <expression (**IN**) of num>

[Stddev ‘:=’] <expression (**IN**) of num> ‘)’

RandomInt

Usage

RandomInt is used to return a random integer number bounded between a lower value and an upper value.

Basic Example

```
 PROC Example()
   !Examples of Random Integers, 5 to 10:
   FOR i FROM 1 TO 10 DO
     ErrWrite\i,"Int: "+ValToStr(RandomInt(5,10)),"";
   ENDFOR
 ENDPROC
```

Return Value

Data Type: Num

The Random integer number between the lower & upper.

Arguments

Lower

Lower Range Limit

DataType: Num

The lower number in the Range .

Upper

Upper Range Limit

DataType: Num

The Upper number in the Range .

Syntax

RandomInt ‘(’

[Lower ‘:=’] <expression (**IN**) of num>

[Upper ‘:=’] <expression (**IN**) of num> ‘)’

RandomUniform

Usage

RandomUniform is used to return a random decimal number bounded between a 0 and 1.

Basic Example

```
 PROC Example()
  !Examples of Random 0 to 1:
  FOR i FROM 1 TO 10 DO
   ErrWrite\i,"Rand: "+ValToStr(RandomUniform()),"";
  ENDFOR
 ENDPROC
```

Return Value

Data Type: Num

The Random decimal number between the 0 & 1.

Syntax

RandomUniform '(' ')'