Quantum Random Number Generator

QRNG

Version 3.1

Software Development Kit

Manual

June 1 2013

Contents

1 Quantum Random Number Generator (QRNG) 3

2 Module Index 5

2.1 Modules . . . 5

3 File Index 7

3.1 File List . . . 7

4 Module Documentation 9

4.1 Enumerated types . . . 9

4.1.1 Detailed Description . . . 9

4.1.2 Enumeration Type Documentation . . . 9

4.1.2.1 QRN_mode . . . 9

4.1.2.2 QRN_security . . . 9

4.1.2.3 QRNReturn . . . 10

4.2 Constructr, destructor and error handling . . . 11

4.2.1 Detailed Description . . . 11

4.2.2 Function Documentation . . . 11

4.2.2.1 PrintErrorCode . . . 11

4.2.2.2 QRN_constr . . . 11

4.2.2.3 QRN_destr . . . 12

4.2.2.4 QRN_GetVersion . . . 12

4.3 Set methods . . . 13

4.3.1 Detailed Description . . . 13

4.3.2 Function Documentation . . . 13

4.3.2.1 QRN_SetSecurityMode . . . 13

4.3.2.2 QRN_SetWorkingMode . . . 13

4.4 Get methods . . . 14

4.4.1 Detailed Description . . . 14

4.4.2 Function Documentation . . . 14

4.4.2.1 QRN_rnd32 . . . 14

4.4.2.2 QRN_rnd32_N . . . 14

4.4.2.3 QRN_rnd64 . . . 15
4.4.2.4 QRN_rnd64_N . . . 15
4.4.2.5 QRN_rndByte . . . 15
4.4.2.6 QRN_rndByte_N . . . 16
4.4.2.7 QRN_unif01 . . . 16
4.4.2.8 QRN_unif01_N . . . 16

5 File Documentation **19**

5.1 libQRN.h File Reference . . . 19
5.1.1 Detailed Description . . . 20
5.1.2 Typedef Documentation . . . 20
5.1.2.1 QRN . . . 20

Chapter 1

Quantum Random Number Generator (QRNG)

Optical Quantum Random Number Generators are a special class of true random data sources, which were successfully applied to cryptography, Monte Carlo numerical simulations and many other fields of mathematics, physics and finance. This class of generators exloits the intrinsical randomness of photonic quantum processes. QRN is made of a CMOS device featuring an array of single photon counting avalanche diodes. The system overcomes the limitations of the commonly used optical QRNG and it is available in three speed-grades of 16 - 32 and 64 MBit/s. Each device was checked against standard statistical test suites and more stringent correlation and bias tests with data streams up to 32 Gbit to prove the quality of the random data generation process. QRN is one of the fastest optical generator currently available which passes some of the most severe test suites for random numbers generators.

IMPORTANT

In order to run the drivers, a set of files should be placed in the same directory as the executable file. The list of the required files is:

```
libQRN.dll                  Software development kit interface
okFrontPanel.dll            Low-level interface
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Enumerated types 9
Constructr, destructor and error handling 11
Set methods 13
Get methods 14

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

libQRN.h
QRN software interface 19

Chapter 4

Module Documentation

4.1 Enumerated types

Enumerations

- enum QRN_mode { SOFT = 0, HARD = 1 << 6 }

  QRN working mode.
- enum QRN_security { MASK_OFF =0, MASK_ON =1 }

  Security mode.
- enum QRNReturn {
  OK = 0, USB_DEVICE_NOT_RECOGNIZED = -1, ELECTRONIC_INTERFACE_NOT_RECOGNIZED =-2, FAILED_FPGA_CONFIGURATION =-3,
  FPGA_USB_DRIVER_FAILURE =-4, OUT_OF_BOUND = -6, MISSING_DLL = -7, EMPTY_BUFFER = -8, NOT_EN_MEMORY = -9, NULL_POINTER = -10, INVALID_OP = -11, UNABLE_CREATE_FILE = -12, UNABLE_READ_FILE = -13, FIRMWARE_NOT_COMPATIBLE =-14, USB_PORT_NOT_EN_POWER = -15, TOO_MUCH_LIGHT = -16,
  QRN_HARDWARE_ERROR = -17, QRN_HARDWARE_ERROR_PRNG_ACTIVE = -18, COMMUNICATION_ERROR = -19 }

  Error table.

4.1.1 Detailed Description

4.1.2 Enumeration Type Documentation

4.1.2.1 enum QRN_mode

QRN working mode.

Select whether the QRN is working in Hardware or Software mode. Software mode means that the random data are transferred to the computer via the USB 2.0 interface. Conversely, the random data are sent to the SMA "Bit stream out" after every low-high logic transition at the SMA "Sync in" input in Hardware mode.

Enumerator

SOFT The random numbers are requested using the SDK functions.

HARD The random numbers are requested using the SMA connectors.

4.1.2.2 enum QRN_security

Security mode.

The output bit stream is masked with the output of a secure Pseudo Random Number Generator (PRNG). The XOR between the Quantum and the PRNG data streams is performed.

Enumerator

MASK_OFF Disable the masking.

MASK_ON Enable the masking.

4.1.2.3 enum QRNReturn

Error table.

Error code returned by the SPC2 functions. Additionally, the state of the device is shown by the LEDs on the front panel.

Left LED	Right LED	Message
Red	Red	Too long sequence of bits equal to 0 or 1
Red	Off	Hardware mode: the bit stream is read too fast for the device
Off	Red	Device illumination failure
Red	Green	USB power error: use a double USB cable or an externally powered USB HUB
Green	Red	Chip error. Contact Micro Photon Devices for assistance
Green	Off	Device ON and idle
Green	Green	Reading data

Enumerator

OK The function is properly called.

USB_DEVICE_NOT_RECOGNIZED The USB device driver has not been properly initialized. Is there any device connected?

ELECTRONIC_INTERFACE_NOT_RECOGNIZED The electronic interface is not able to communicate with the computer.

FAILED_FPGA_CONFIGURATION The configuration of the on-board FPGA device failed.

FPGA_USB_DRIVER_FAILURE The FPGA firmware is corrupted.

OUT_OF_BOUND One or more parameters passed to the function are outside the valid boundaries.

MISSING_DLL One or more SPC2 libraries are missing.

EMPTY_BUFFER An emtpy buffer image has been provided to the function.

NOT_EN_MEMORY Not enaugh memory is available to start the camera.

NULL_POINTER A null pointer has been provided to the function.

INVALID_OP The required function can not be executed. The device is probably busy.

UNABLE_CREATE_FILE An output file can not be created.

UNABLE_READ_FILE The provided file can not be accessed.

FIRMWARE_NOT_COMPATIBLE The camera firmare is not compatble with the current software.

USB_PORT_NOT_EN_POWER Not enough power for the camera. Change USB port and check the length of the USB cable.

TOO_MUCH_LIGHT Too much light was detected by the camera. The protection mechanism has been enable.

QRN_HARDWARE_ERROR An hardware error was detected on the QRN. If the error persists contact MPD.

QRN_HARDWARE_ERROR_PRNG_ACTIVE An hardware error was detected on the QRN. Since the security feature is enabled, data is provided only by the KISS PRNG. If the error persists contact MPD.

COMMUNICATION_ERROR Communication error during readout. Check USB connection.

4.2 Constructr, destructor and error handling

Functions

- DllQRNExport QRNReturn QRN_constr (QRN ∗q, QRN_mode Qm, QRN_security Qs, char ∗Device_ID)

  Constructor.
- DllQRNExport QRNReturn QRN_destr (QRN q)

  Destructor.
- DllQRNExport QRNReturn QRN_GetVersion (QRN q, double ∗Firmware_Version, double ∗Software_Version)

  Get the firmware and software versions.
- DllQRNExport void PrintErrorCode (FILE ∗fout, const char ∗FunName, const QRNReturn retcode)

  Print an error message.

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 DllQRNExport void PrintErrorCode (FILE ∗ *fout*, const char ∗ *FunName*, const QRNReturn *retcode*)

Print an error message.

All the SDK functions return an error code to inform the user whether the issued command was successfully executed or not. The result of the execution of a function can be redirect to a text file by providing a valid file pointer.

Parameters

fout	Output text file
FunName	Additional text to define the warning/error. Usually the name of the calling function is provided.
retcode	Error code returned by a SDK command

4.2.2.2 DllQRNExport QRNReturn QRN_constr (QRN ∗ *q*, QRN_mode *Qm*, QRN_security *Qs*, char ∗ *Device_ID*)

Constructor.

It allocates a memory block to contain all the information and buffers required by the QRN. If multiple devices are connected to the computer, a unique Device ID should be provided to correctly identify the camera. The camera ID can be found in the camera documentation (9 numbers and a letter) and it is printed on the screen during initialization. An empty string is accepted too. In this case, the devices will be connected in the order which is printed on the screen. The random bits are either send to the computer via the USB interface or to the output SMA port by setting the required operation mode (QRN_mode).

Parameters

q	Pointer to the QRN handle
Qm	QRN output mode
Qs	Enable or disable the XOR with a pseudo random number generator
Device_ID	Unique ID to identify the connected device

Returns

OK
INVALID_OP q points to an occupied memory location
USB_DEVICE_NOT_RECOGNIZED A valid device has not been recognized
FIRMWARE_NOT_COMPATIBLE The SDK and Firmware versions are not compatible
NOT_EN_MEMORY There is not enough memory to run the QRN

MISSING_DLL One or more QRN libraries are missing

4.2.2.3 DllQRNExport QRNReturn QRN_destr (QRN *q*)

Destructor.

It deallocates the memory block which contains all the information and buffers required by the QRN. **WARNING** the user must call the destructor before the end of the program to avoid memory leakages.

Parameters

q	QRN handle

Returns

OK
NULL_POINTER The provided device handle points to an empty memory location

4.2.2.4 DllQRNExport QRNReturn QRN_GetVersion (QRN *q*, double * *Firmware_Version*, double * *Software_Version*)

Get the firmware and software versions.

Get the firmware and software versions.

Parameters

q	QRN handle
Firmware_Version	Version of the firmware
Software_Version	Version of the software

Returns

OK
NULL_POINTER The provided device handle or one of the variables point to an empty memory location

4.3 Set methods

Functions

- DllQRNExport QRNReturn QRN_SetSecurityMode (QRN q, QRN_security mode)

  Set the security mode of the device Set the security mode of the device.
- DllQRNExport QRNReturn QRN_SetWorkingMode (QRN q, QRN_mode mode)

  Set the working mode of the device Set the working mode of the device.

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 DllQRNExport QRNReturn QRN_SetSecurityMode (QRN *q*, QRN_security *mode*)

Set the security mode of the device Set the security mode of the device.

When the security mode is MASK_ON, the quantum bit stream is masked with the output of a pseundo PRG to enhance the security level.

Parameters

q	QRN handle
mode	New security mode

Returns

OK
NULL_POINTER The provided device handle points to an empty memory location

4.3.2.2 DllQRNExport QRNReturn QRN_SetWorkingMode (QRN *q*, QRN_mode *mode*)

Set the working mode of the device Set the working mode of the device.

When the device is in hardware (HARD) mode, the random data cannot be transferred to the host computer via USB. In this case, an INVALID_OP exception is returned.

Parameters

q	QRN handle
mode	New working mode

Returns

OK
NULL_POINTER The provided device handle points to an empty memory location

4.4 Get methods

Functions

- DllQRNExport QRNReturn QRN_unif01 (QRN q, double *num)

  Generate a uniform number [0,1].
- DllQRNExport QRNReturn QRN_unif01_N (QRN q, double *MemArray, int NElements)

  Generate an array of uniform numbers [0,1].
- DllQRNExport QRNReturn QRN_rnd32 (QRN q, UInt32 *num)

  Generate a 32-bit unsigned integer number.
- DllQRNExport QRNReturn QRN_rnd32_N (QRN q, UInt32 *MemArray, int NElements)

  Generate an array of 32-bit unsigned integer numbers.
- DllQRNExport QRNReturn QRN_rnd64 (QRN q, UInt64 *num)

  Generate a 64-bit unsigned integer number.
- DllQRNExport QRNReturn QRN_rnd64_N (QRN q, UInt64 *MemArray, int NElements)

  Generate an array of 64-bit unsigned integer numbers.
- DllQRNExport QRNReturn QRN_rndByte (QRN q, byte *num)

  Generate a 8-bit unsigned integer number.
- DllQRNExport QRNReturn QRN_rndByte_N (QRN q, byte *MemArray, int NElements)

  Generate an array of 8-bit unsigned integer numbers.

4.4.1 Detailed Description

4.4.2 Function Documentation

4.4.2.1 DllQRNExport QRNReturn QRN_rnd32 (QRN *q*, UInt32 * *num*)

Generate a 32-bit unsigned integer number.

Parameters

q	QRN handle
num	Pointer to the number

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested by the SDK functions.

4.4.2.2 DllQRNExport QRNReturn QRN_rnd32_N (QRN *q*, UInt32 * *MemArray*, int *NElements*)

Generate an array of 32-bit unsigned integer numbers.

The array must be previously allocated by the user.

Parameters

q	QRN handle
MemArray	Pointer to the array of 32-bit unsigned integer numbers.
NElements	Number of elements of the array

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested
OUT_OF_BOUND The requested number of values is outside the valid range

4.4.2.3 DllQRNExport QRNReturn QRN_rnd64 (QRN *q*, UInt64 * *num*)

Generate a 64-bit unsigned integer number.

Parameters

q	QRN handle
num	Pointer to the number

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested by the SDK functions.

4.4.2.4 DllQRNExport QRNReturn QRN_rnd64_N (QRN *q*, UInt64 * *MemArray*, int *NElements*)

Generate an array of 64-bit unsigned integer numbers.

The array must be previously allocated by the user.

Parameters

q	QRN handle
MemArray	Pointer to the array of 64-bit unsigned integer numbers.
NElements	Number of elements of the array

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested
OUT_OF_BOUND The requested number of values is outside the valid range

4.4.2.5 DllQRNExport QRNReturn QRN_rndByte (QRN *q*, byte * *num*)

Generate a 8-bit unsigned integer number.

Parameters

q	QRN handle
num	Pointer to the number

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested by the SDK functions.

4.4.2.6 DllQRNExport QRNReturn QRN_rndByte_N (QRN *q*, byte * *MemArray*, int *NElements*)

Generate an array of 8-bit unsigned integer numbers.

The array must be previously allocated by the user.

Parameters

q	QRN handle
MemArray	Pointer to the array of 8-bit unsigned integer numbers.
NElements	Number of elements of the array

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested
OUT_OF_BOUND The requested number of values is outside the valid range

4.4.2.7 DllQRNExport QRNReturn QRN_unif01 (QRN *q*, double * *num*)

Generate a uniform number [0,1].

The double-precision floating point number is obtained from a 32 bit integer. The double-precision floating point numbers have 52 bit mantissa in the IEEE 754 standard, but only 32 bit are used. The numbers are uniformly spaced with a step of 1/2^32.

Parameters

q	QRN handle
num	Pointer to the number

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested by the SDK functions.

4.4.2.8 DllQRNExport QRNReturn QRN_unif01_N (QRN *q*, double * *MemArray*, int *NElements*)

Generate an array of uniform numbers [0,1].

Generate an array of double-precision floating point numbers. The array must be previously allocated by the user.

See Also

QRN_unif01()

Parameters

q	QRN handle
MemArray	Pointer to the array of double-precision numbers.
NElements	Number of elements of the array

Returns

OK
NULL_POINTER The provided device handle or parameter points to an empty memory location
INVALID_OP The device has been initialized in hardware mode. No random data can be requested
OUT_OF_BOUND The requested number of values is outside the valid range by the SDK functions.

Chapter 5

File Documentation

5.1 libQRN.h File Reference

QRN software interface.

Typedefs

- typedef unsigned long long **UInt64**
- typedef unsigned **UInt32**
- typedef unsigned short **UInt16**
- typedef struct _QRN * QRN

  QRN handle.

Enumerations

- enum QRN_mode { SOFT = 0, HARD = 1 << 6 }

  QRN working mode.
- enum QRN_security { MASK_OFF =0, MASK_ON =1 }

  Security mode.
- enum QRNReturn {
  OK = 0, USB_DEVICE_NOT_RECOGNIZED = -1, ELECTRONIC_INTERFACE_NOT_RECOGNIZED =-2, FAILED_FPGA_CONFIGURATION =-3,
  FPGA_USB_DRIVER_FAILURE =-4, OUT_OF_BOUND = -6, MISSING_DLL = -7, EMPTY_BUFFER = -8, NOT_EN_MEMORY = -9, NULL_POINTER = -10, INVALID_OP = -11, UNABLE_CREATE_FILE = -12, UNABLE_READ_FILE = -13, FIRMWARE_NOT_COMPATIBLE =-14, USB_PORT_NOT_EN_POWER = -15, TOO_MUCH_LIGHT = -16,
  QRN_HARDWARE_ERROR = -17, QRN_HARDWARE_ERROR_PRNG_ACTIVE = -18, COMMUNICATION_ERROR = -19 }

  Error table.

Functions

- DllQRNExport QRNReturn QRN_constr (QRN *q, QRN_mode Qm, QRN_security Qs, char *Device_ID)

  Constructor.
- DllQRNExport QRNReturn QRN_destr (QRN q)

  Destructor.
- DllQRNExport QRNReturn QRN_GetVersion (QRN q, double *Firmware_Version, double *Software_Version)

Get the firmware and software versions.

- DllQRNExport void PrintErrorCode (FILE *fout, const char *FunName, const QRNReturn retcode)

  Print an error message.
- DllQRNExport QRNReturn QRN_SetSecurityMode (QRN q, QRN_security mode)

  Set the security mode of the device Set the security mode of the device.
- DllQRNExport QRNReturn QRN_SetWorkingMode (QRN q, QRN_mode mode)

  Set the working mode of the device Set the working mode of the device.
- DllQRNExport QRNReturn QRN_unif01 (QRN q, double *num)

  Generate a uniform number [0,1].
- DllQRNExport QRNReturn QRN_unif01_N (QRN q, double *MemArray, int NElements)

  Generate an array of uniform numbers [0,1].
- DllQRNExport QRNReturn QRN_rnd32 (QRN q, UInt32 *num)

  Generate a 32-bit unsigned integer number.
- DllQRNExport QRNReturn QRN_rnd32_N (QRN q, UInt32 *MemArray, int NElements)

  Generate an array of 32-bit unsigned integer numbers.
- DllQRNExport QRNReturn QRN_rnd64 (QRN q, UInt64 *num)

  Generate a 64-bit unsigned integer number.
- DllQRNExport QRNReturn QRN_rnd64_N (QRN q, UInt64 *MemArray, int NElements)

  Generate an array of 64-bit unsigned integer numbers.
- DllQRNExport QRNReturn QRN_rndByte (QRN q, byte *num)

  Generate a 8-bit unsigned integer number.
- DllQRNExport QRNReturn QRN_rndByte_N (QRN q, byte *MemArray, int NElements)

  Generate an array of 8-bit unsigned integer numbers.

5.1.1 Detailed Description

QRN software interface. This C header containes all the functions to operate the QRN in user defined applications.

5.1.2 Typedef Documentation

5.1.2.1 typedef struct _QRN* QRN

QRN handle.

Pointer to the QRN data structure.