Quantis Library Documentation

id Quantique

April 11, 2007

Contents

1 Introduction

The Quantis library can be used to access Quantis Quantum Random Number Generator. The library API is the same for the PCI and the USB library and is OS independent. This documentation has been automatically generated from

the library sourcecode, and contains the complete header for the library, as well as examples of using the library.

2 Quantis library

2.1 Preamble

We define the `QUANTIS_H` symbol to avoid multiple inclusion of the header file, and export the symbols as `C` symbols when the library is used in C++ mode.

```
#ifndef QUANTIS_H
#define QUANTIS_H
```

We are at version 1.1.

```
#define QUANTIS_LIB_VERSION 11

#ifdef __cplusplus
extern "C" {
#endif
```

3 Informative functions

These functions return information about the number of cards present, the version of the hardware and of the software, as well as number of modules present on a board.

If the PCI library is used, the card number parameter refers to the number of the Quantis PCI card and if the USB library is used the card number parameter refers to the number of the Quantis USB device.

The Quantis PCI library is named `libquantis`. The Quantis USB library is named `libquantis-usb`.

3.1 quantisCount - return the number of Quantis boards

Returns the number of cards installed in the system.

If an error occured, the function returns 0 (we assume that no card are installed in the system).

```
int quantisCount(void);
```

3.2 quantisBoardVersion - get the Quantis board version

This function takes the card number as a parameter and returns the version of the board, or 0 (which is an invalid version number) when an error occurred. The version is a 4 bytes number in the format:

```
[year][month][day][r]
31..24 23..16 15..8 7..0
```

where `r` indicates the release number of the day.

```
int quantisBoardVersion(int cardNumber);
```

3.3 quantisLibVersion - get the Quantis library version

This function returns the version of the library. The version is a decimal number composed of the major number and the minor version: `version = major * 10 + minor`.

```
int quantisLibVersion(void);
```

3.4 quantisDriverVersion - get the version of the Quantis driver

This function returns the version of the Quantis driver. The version is a decimal number composed of the major number and the minor version: `version = major * 10 + minor`.

It returns 0 (which is an invalid version number) when an error occured or no card could be found.

```
int quantisDriverVersion(void);
```

4 Board functions

These functions are used to reset, enable, disable and get the status of modules on a Quantis board.

4.1 quantisBoardReset (lowlevel) - reset the Quantis board

This function takes the card number as a parameter. It resets the board, setting all values to their initial state and enabling all the modules present on the card.

The function returns 0 on success, and a negative value on error.

Normally, this function does not need to be called, as the board is reset on driver loading (on unix systems) or dll loading (on windows systems).

```
int quantisBoardReset(int cardNumber);
```

4.2 quantisGetModules - detect the modules on a board

This function takes the card number as a parameter. The return value is a bitmask of the modules present on the card. Bit 0 is set when module 0 is present, bit 1 when module 1 is present, bit 2 when module 2 is present and bit 3 when module 3 is present. For example, a return value of `101` in binary (5 in decimal) shows that module 0 and module 2 are present.

If the card is not present or an error occured, a bitmask of 0 is returned (no modules installed).

```
int quantisGetModules(int cardNumber);
```

4.3 quantisModulesStatus (lowlevel) - get the status of the modules on a board

This function takes the card number as first parameter.

This function returns the status of the modules on the card as a bitmask similar to the second argument to `quantisModulesReset`. A set bit signals the module is in a correct state. It returns -1 on error.

This function returns only the status of the modules that are currently enabled.

```
int quantisModulesStatus(int cardNumber);
```

4.4 quantisModulesReset (lowlevel) - reset some modules on a board

This function takes the card number as first parameter, and the mask of the modules to reset as second parameter. The mask is similar to the bitmask returned by the `quantisGetModules` function, bit 0 corresponds to module 0, bit 1 to module 1, bit 2 to module 2, bit 3 to module 3. The modules for which the bitmask is set are reset.

This function returns 0 on success, and -1 if an error occured.

This function first disables the modules by calling `quantisModulesDisable` and then enables the modules by calling `quantisModulesEnable`. Thus, only the reset modules will be enabled after a call to `quantisModulesReset`.

```
int quantisModulesReset(int cardNumber, int moduleMask);
```

4.5 quantisModulesEnable (lowlevel) - enable some modules on a board

This function takes the card number as first parameter, and the mask of the modules to enable as second parameter. The mask is similar to the bitmask given as second argument to `quantisModulesReset`.

This function returns 0 on success, and -1 if an error occured.

After a call to `quantisModulesEnable`, reading from the Quantis board using `quantisRead` will also read from the enabled modules in addition to the already enabled modules.

```
int quantisModulesEnable(int cardNumber, int moduleMask);
```

4.6 quantisModulesDisable (lowlevel) - disable some modules on a board

This function takes the card number as first parameter, and the mask of the modules to disable as second parameter. The mask is similar to the bitmask given as second argument to `quantisModulesReset`.

This function returns 0 on success, and -1 if an error occured.

After a call to `quantisModulesDisable`, reading from the Quantis board using `quantisRed` will not read from the disabled modules anymore.

```
int quantisModulesDisable(int cardNumber, int moduleMask);
```

5 Read function

This is the most important function that reads random data from the Quantis board.

5.1 quantisRead - read from a Quantis board

This function takes the card number as first parameter, a pointer to a destination buffer in memory as second argument, and the number of bytes to read from the board as third parameter.

This function returns the number of bytes read on success, and -1 on error. In case of success, it always returns the exact number of bytes requested.

Please note with PCI version that reading a single random byte is inefficient, as the FIFO of the PCI board is read in 32 bytes chunk. For high-throughput, multiples of 4 bytes should be read from the PCI board or the USB device.

```
int quantisRead(int cardNumber, void *buffer, unsigned int size);
```

5.2 Postamble

Close the C++ `extern ''C''` environment.

```
#ifdef __cplusplus
}
#endif

#endif // QUANTIS_H
```

6 Examples for the Quantis PCI library

This part shows examples of applications using the Quantis library. It is automatically generated from the `examples.c` file and contains the complete sourcecode.

```
#include <stdio.h>
#include "quantis.h"
```

6.1 Getting information about Quantis boards

This example shows how the Quantis library can be used to get information about the present Quantis PCI boards.

First, we define a function to print module bitmasks. If the bit corresponding to a module is set in the bitmask, its name is printed on the standard output.

```
void quantis_example_print_bitmask(int bitmask) {
  int i;
  for (i = 0; i < 4; i++) {
    if (bitmask & (1 << i)) {
      printf("Bit for module %d is set\n", i);
    }
  }
}
```

The main function of this example first prints some information about the library version and the driver version. It then gets the number of installed Quantis PCI boards, and cycles through the boards, printing their board version, the names of the enabled modules and the status of the enabled modules.

```
void quantis_example_information(void) {
  int count;
  int i;
  int lib_version, driver_version;
```

Print the version information. If we can't get the driver version, we print an error message and quit.

```
  lib_version = quantisLibVersion();
  printf("Quantis Library Version: %d.%d\n", lib_version / 10, lib_version % 10);
  driver_version = quantisDriverVersion();
  if (driver_version == 0) {
    printf("Could not get the Quantis driver version.\n");
    printf("This could mean that no card is installed.\n");
    return;
  } else {
    printf("Quantis Driver Version: %d.%d\n", driver_version / 10,
      driver_version % 10);
  }

  count = quantisCount();
  printf("Quantis information example: found %d Quantis PCI boards\n", count);
```

Now we print information about each Quantis PCI board. First we get the board version, and jump to the next board if we get an error. Then we get the modules mask, which shows us which modules are present on the PCI board. We then print the current status of each enabled module if some modules are present.

```
  for (i = 0; i < count; i++) {
    int modules_mask;
    int status_mask;
    int board_version;

    board_version = quantisBoardVersion(i);
    if (board_version == 0) {
      printf("Could not get the board version of the Quantis PCI board nr. %d.\n",
        i);
      continue;
    } else {
      printf("Quantis PCI board nr %d, board version: %.4x\n", i, board_version);
    }

    modules_mask = quantisGetModules(i);
    if (modules_mask == 0) {
      printf("Quantis PCI board nr %d: no modules are installed\n", i);
      continue;
    } else {
      printf("Quantis PCI board nr %d: showing present modules\n", i);
```

```
      quantis_example_print_bitmask(modules_mask);
    }

    status_mask = quantisModulesStatus(i);
    if (status_mask < 0) {
      printf("Quantis PCI board nr %d: could not get the modules status\n", i);
      continue;
    } else {
      printf("Quantis PCI board nr %d: showing the status of enabled modules\n", i);
      quantis_example_print_bitmask(status_mask);
    }
  }
}
```

6.2 Reading random data from a Quantis PCI board

In this example, we read random data from the first Quantis PCI board present in the system.

```
void quantis_example_read(void) {
  unsigned char buf[4096];
  int bytes;

  printf("Quantis read example: trying to read 4096 bytes from the first PCI board\n");
  bytes = quantisRead(0, buf, 4096);
  if (bytes < 0) {
    printf("Quantis read example: got an error while trying to read 4096 bytes\n");
  } else {
    printf("Quantis read example: read %d bytes from the first PCI board\n", bytes);
  }
}
```

6.3 Test main routine

This is the main routine for the test program, which calls all the test routines.

```
int main(void) {
  quantis_example_information();
  quantis_example_read();
  return 0;
}
```