libpyramicio

1.0

Generated by Doxygen 1.8.12

Contents

Chapter 1

Documentation for libpyramicio

1.1 What is libpyramicio ?

This library is an abstraction layer for the Pyramic array -made at LCAV (EPFL)- input/output functions. It enables the use of the Pyramic array by designing software against an existing hardware design without having to use Altera Quartus Prime tools, or recompiling the application at each change of the design in VHDL.

1.2 How to use it ?

In order to use the library, one just has to include the <pyramicio.h> file, then initialize a Pyramic object through the pyramicInitializePyramic() function.

All the usable functions are documented in the pyramicio.h file reference in this documentation.

Note that programs that use the Pyramic have to be run as root, because the library is using direct references to memory areas that are reserved for the system.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 inputBuffer Struct Reference

This structure represents an input buffer, which direction is the microphone array towards the memory.

`#include <pyramicio.h>`

Public Attributes

- int microphoneCount

  How many microphones the Pyramic array has.
- uint32_t totalSampleCount

  How many samples the buffer contains.
- uint32_t samplesPerMic

  How many samples are found for each microphone in the buffer.
- int16_t * samples

  The actual samples, organized the RIFF way.

4.1.1 Detailed Description

This structure represents an input buffer, which direction is the microphone array towards the memory.

4.1.2 Member Data Documentation

4.1.2.1 microphoneCount

`int inputBuffer::microphoneCount`

How many microphones the Pyramic array has.

4.1.2.2 samples

```
int16_t* inputBuffer::samples
```

The actual samples, organized the RIFF way.

4.1.2.3 samplesPerMic

```
uint32_t inputBuffer::samplesPerMic
```

How many samples are found for each microphone in the buffer.

4.1.2.4 totalSampleCount

```
uint32_t inputBuffer::totalSampleCount
```

How many samples the buffer contains.

The documentation for this struct was generated from the following file:

- pyramicio.h

4.2 outputBuffer Struct Reference

This structure represents an output buffer, which direction is the memory towards the FPGA CODEC.

```
#include <pyramicio.h>
```

Public Attributes

- uint32_t **baseAddress**
- uint32_t **length**
- int16_t * **samples**

4.2.1 Detailed Description

This structure represents an output buffer, which direction is the memory towards the FPGA CODEC.

This CODEC is configured to work at 48000 Hz, hence the sampling rate of the injected audio *has* to be 48000 Hz.

The documentation for this struct was generated from the following file:

- pyramicio.h

4.3 pyramic Struct Reference

Structure contanining the addresses used by the library internals .

`#include <pyramicio.h>`

Public Attributes

- void * **h2f_lw_axi_master**
- size_t **h2f_lw_axi_master_span**
- size_t **h2f_lw_axi_master_ofst**
- void * **fpga_SPI_System**
- void * **fpga_Output_Controller**
- int **fd_dev_mem**
- void * **reserved_memory**
- void * **output_memory**
- int **captureDuration**

4.3.1 Detailed Description

Structure contanining the addresses used by the library internals .

The documentation for this struct was generated from the following file:

- pyramicio.h

Chapter 5

File Documentation

5.1 pyramicio.h File Reference

A library that allows an easy access to the Pyramic array.

```
#include <inttypes.h>
#include <unistd.h>
```

Classes

- struct inputBuffer

  This structure represents an input buffer, which direction is the microphone array towards the memory.
- struct outputBuffer

  This structure represents an output buffer, which direction is the memory towards the FPGA CODEC.
- struct pyramic

  Structure contanining the addresses used by the library internals .

Macros

- #define **SRC_BEAMFORMER** 0
- #define **SRC_MEMORY** 1

Functions

- struct pyramic ∗ pyramicInitializePyramic ()

  Initializes the Pyramic array and returns a reference to the associated Pyramic object.
- void pyramicDeinitPyramic (struct pyramic ∗p)

  Closes the file descriptors assoiated with the Pyramic and frees the reserved memory resources.
- struct inputBuffer ∗ pyramicGetInputBuffer (struct pyramic ∗p, int bufferHalf)

  Gets the current input buffer.
- int pyramicGetCurrentBufferHalf (struct pyramic ∗p)

  Gets the number of the half on which the Pyramic is currently recording samples.
- struct outputBuffer ∗ pyramicAllocateOutputBuffer (struct pyramic ∗p, uint32_t lengthInSamples)

Allocates memory as a buffer to output samples.

- void pyramicDeallocateOutputBuffer (struct pyramic *p, struct outputBuffer *outputBuffer)

  Sets the Pyramic output buffer to be the specified address space.
- int pyramicStartCapture (struct pyramic *p, int bufferLengthInSeconds)

  Starts a continuous capture on the Pyramic array.
- int pyramicFixedLengthCapture (struct pyramic *p, int durationInSeconds)

  Starts a fixed length capture on the Pyramic array.
- int pyramicStopCapture (struct pyramic *p)

  Stops the ongoing capture on the Pyramic array at the end of the current sample.
- int pyramicSelectOutputSource (struct pyramic *p, int source)

  Selects if the output samples come from the Beamformer or a software buffer.
- int pyramicSetOutputBuffer (struct pyramic *p, struct outputBuffer *outputBuffer)

  Sets the Pyramic's output buffer as the designated one.

5.1.1 Detailed Description

A library that allows an easy access to the Pyramic array.

This library is compiled using headers derived from the VHDL code available at: http://github.com/lcav/pyramic.git. In order to compile this library, one has to run the "headers_rbf.sh" file that can be found in the toplevel MIC_ARRAY directory to provide the Quartus generated header files.

The pyramicio.h file gives access to the API provided by libpyramicio. It enables the use of a Pyramic array with an abstraction layer that removes the hassle of the FPGA addresses. This enables programming applications that use the Pyramic array against an existing design without using the Quartus Prime tools.

Author

Corentin Ferry

Date

December 2016

See also

https://github.com/cferr/pyramic.git

5.1.2 Function Documentation

5.1.2.1 pyramicAllocateOutputBuffer()

```
struct outputBuffer* pyramicAllocateOutputBuffer (
            struct pyramic * p,
            uint32_t lengthInSamples )
```

Allocates memory as a buffer to output samples.

Parameters

p	The Pyramic object structure on which the function is executed.
lengthInSamples	The nummber of samples that the output buffer will hold. Note that the output frequency is 48000 Hz.

5.1.2.2 pyramicDeallocateOutputBuffer()

```
void pyramicDeallocateOutputBuffer (
            struct pyramic * p,
            struct outputBuffer * outputBuffer )
```

Sets the Pyramic output buffer to be the specified address space.

Parameters

p	The Pyramic object structure on which the function is executed.
outputBuffer	The output buffer that has to be freed.

5.1.2.3 pyramicDeinitPyramic()

```
void pyramicDeinitPyramic (
            struct pyramic * p )
```

Closes the file descriptors assoiated with the Pyramic and frees the reserved memory resources.

Parameters

p	The Pyramic object structure on which the function is executed.

5.1.2.4 pyramicFixedLengthCapture()

```
int pyramicFixedLengthCapture (
            struct pyramic * p,
            int durationInSeconds )
```

Starts a fixed length capture on the Pyramic array.

Parameters

p	The Pyramic object structure on which the function is executed.
durationInSeconds	The duration of the capture. After the capture, you will be able to get the samples through the pyramicGetInputBuffer() function.

5.1.2.5 pyramicGetCurrentBufferHalf()

```
int pyramicGetCurrentBufferHalf (
            struct pyramic * p )
```

Gets the number of the half on which the Pyramic is currently recording samples.

The other half can be safely used for processing the signal.

Parameters

p	The Pyramic object structure on which the function is executed.

5.1.2.6 pyramicGetInputBuffer()

```
struct inputBuffer* pyramicGetInputBuffer (
            struct pyramic * p,
            int bufferHalf )
```

Gets the current input buffer.

Parameters

p	The Pyramic object structure on which the function is executed.
bufferHalf	If this parameter is 0, the samples start at the beginning of the buffer (thus giving you access to the first half and the second half as well). If it is 1, the samples start at the beginning of the second half of the buffer. This parameter is useful for continuous captures where it is safe to use a single half of the buffer at a time.

5.1.2.7 pyramicInitializePyramic()

```
struct pyramic* pyramicInitializePyramic ( )
```

Initializes the Pyramic array and returns a reference to the associated Pyramic object.

This initialization is exclusive: only one thread can have control over the Pyramic array at the same time.

5.1.2.8 pyramicSelectOutputSource()

```
int pyramicSelectOutputSource (
            struct pyramic * p,
            int source )
```

Selects if the output samples come from the Beamformer or a software buffer.

Parameters

p	The Pyramic object structure on which the function is executed.
source	Either SRC_BEAMFORMER (not implemented yet, gives silence) or SRC_MEMORY (the pyramic then takes its input from an output buffer in the DDR3). It is recommended to set the output buffer
	address through pyramicSetOutputBuffer() before calling this function with SRC_MEMORY.

5.1.2.9 pyramicSetOutputBuffer()

```
int pyramicSetOutputBuffer (
            struct pyramic * p,
            struct outputBuffer * outputBuffer )
```

Sets the Pyramic's output buffer as the designated one.

Parameters

p	The Pyramic object structure on which the function is executed.
outputBuffer	An output buffer that has been allocated with pyramicAllocateOutputBuffer().

5.1.2.10 pyramicStartCapture()

```
int pyramicStartCapture (
            struct pyramic * p,
            int bufferLengthInSeconds )
```

Starts a continuous capture on the Pyramic array.

Parameters

p	The Pyramic object structure on which the function is executed.
bufferLengthInSeconds	The duration of the sample buffer. Note that the sample buffer is divided into two halves, and you can safely read and write into each half while it is not being processed, using pyramicGetCurrentBufferHalf(). The buffer has to be long enough so that half of it can be entirely processed while the other half is under capture. You can get the capture buffers through the pyramicGetInputBuffer() function.

5.1.2.11 pyramicStopCapture()

```
int pyramicStopCapture (
            struct pyramic * p )
```

Stops the ongoing capture on the Pyramic array at the end of the current sample.

Parameters

p	The Pyramic object structure on which the function is executed.

Index