DataConverter

1.0

Generated by Doxygen 1.7.4

Tue Apr 19 2011 00:04:10

Contents

Chapter 1

User Manual

1.1 Description

DataConverter converts Agilent and Lecroy files into the format used by CUDA Correlation Attacker.

1.2 How to compile

1.2.1 Compile DataConverter

Run the following command from the main folder (the folder where Makefile is):

```
make
```

If everything goes fine a binary executable file named *converter* is created.

1.2.2 Compile documentation

Run the following command from the main folder:

```
doxygen documentation.cfg
```

Open doc/html/index.html to read html documentation.
In doc/latex, run:

```
make
```

in order to create pdf documentation file (*doc/latex/refman.pdf*).

1.3 How to run

From the main folder, run the following command:

```
./converter <source1> <source2>
```

- *source1:* is the full path of the file containing a list of hexadecimal values
- *source2:* is the full path of the configuration file used during the conversion

See section 1.4 for more details.

1.3.1 Exit codes

- 0: everything goes fine
- 1: command line error
- 2: setting files parsing error

1.4 Configuration

The configuration file must contain lines in this format:

- the couple: <key>=<value>
- the special key: <file_list:> followed by a full path list of the Agilent or Lecroy files
- blank lines and any character following # are ignored

The following values must be set in the configuration file

- *output_trace_length:* the length of the trace to save in number of samples
- *output_trace_offset:* the number of samples to ignore during the conversion
- *output_traces_per_file:* the number of traces to save for each output file
- *output_path:* the location where the output file will be saved
- *output_name:* the name of the output file to save
- *input_format:* the format of the input files (Lecroy, Agilent, txt)
- *file_list:* this special key must be followed by a list of full path of the Agilent or Lecroy files

1.5 Output file

The converter generates one or more output files (it depends on the configuration settings).

An output file contains:

- a header with informations about:
  - the number of traces
  - the number of samples per trace
  - the format of each sample
  - the length of a plain/cipher text in byte
- a list of traces with a plain/cipher text attached to each trace

Both the traces and the plain/cipher texts are saved in binary format.

Here an example:

```
--- header ---
number of traces [uint32]
number of samples per trace [uint32]
sample format [char: b for int8, f for float, d for double]
length of a plain/cipher text in byte [uint8]
--- trace 1 ----
trace [in binary format]
plain/cipher text [in binary format]
--- trace 2 ----
.
.
.

--- trace n ----
```

1.5.1 Sample formats

The samples of a trace could be of four formats. The converter saves this information in the output header using a different character for each format.

- *int8* saved as 'b'
- *int16* saved as 'c'
- *float* saved as 'f'
- *double* saved as 'd'

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 AgilentFileHeaderTag Struct Reference

The type used to represent the header of an Agilent file.

`#include <common.h>`

Public Attributes

- char **cookie** [2]
- char **version** [2]
- unsigned int **fileSize**
- unsigned int **numberOfWaveforms**

4.1.1 Detailed Description

The type used to represent the header of an Agilent file.

The struct represents the equivalent header of an Agilent file

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

- common.h

4.2 AgilentWaveformDataHeaderTag Struct Reference

The type used to represent the data header of each waveform in an Agilent file.

`#include <common.h>`

Public Attributes

- unsigned int **HeaderSize**
- unsigned short **BufferType**
- unsigned short **BytesPerPoint**
- unsigned int **BufferSize**

4.2.1 Detailed Description

The type used to represent the data header of each waveform in an Agilent file.

The struct represents the equivalent data header of a waveform in an Agilent file

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

- common.h

4.3 AgilentWaveformHeaderTag Struct Reference

The type used to represent the header of each waveform in an Agilent file.

`#include <common.h>`

Public Attributes

- unsigned int **HeaderSize**
- unsigned int **WaveformType**
- unsigned int **NWaveformBuffers**
- unsigned int **Points**
- unsigned int **Count**
- float **XDisplayRange**
- double **XDisplayOrigin**
- double **XIncrement**
- double **XOrigin**
- unsigned int **XUnits**
- unsigned int **YUnits**
- char **Date** [DATE_TIME_STRING_LENGTH]
- char **Time** [DATE_TIME_STRING_LENGTH]
- char **Frame** [FRAME_STRING_LENGTH]
- char **WaveformLabel** [SIGNAL_STRING_LENGTH]
- double **TimeTag**
- unsigned int **SegmentIndex**

4.3.1 Detailed Description

The type used to represent the header of each waveform in an Agilent file.

The struct represents the equivalent header of a waveform in an Agilent file

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

- common.h

4.4 FileConfigReader Class Reference

This class parses a converter configuration file.

`#include <FileConfigReader.h>`

Public Member Functions

- FileConfigReader (const string &file_name)

  Creates a new object of the class.
- ~FileConfigReader ()

  Class destructor.
- UINTN getOutputTraceLength ()

  Returns the length of the trace to save.
- UINTN getOutputTraceOffset ()

  Returns the number of samples to ignore during the conversion.
- UINTN getOutputTracePerFile ()

  Returns the number of traces to save in any output file.
- string getOutputPath ()

  Returns the location where the output files will be saved.
- string getOutputName ()

  Returns the name of the output file to save.
- int getInputFormat ()

  Returns the format of the input files (Lecroy, Agilent, txt).
- vector< string > getFileList ()

  Returns the list of the input files to convert with full path.
- bool keyExists (const string &key) const

  Checks if a key exists in the list of parsed keys.
- template<typename ValueType >
  ValueType getValueOfKey (const string &key, ValueType const &defaultValue=ValueType()) const

  Returns the value of a parsed key converted in ValueType.

4.4.1 Detailed Description

This class parses a converter configuration file.

The object created contains all the configuration settings used by the FileReader Class to parse Lecroy/Agilent files

and used by the Converter to create the output files.

The configuration file must contain lines in this format:

- the couple: keyValue = value
- the special key *'file_list*:' followed by a full path list of the Lecroy/Agilent files
- a comment line starts with '#'

4.4.2 Constructor & Destructor Documentation

4.4.2.1 FileConfigReader::FileConfigReader (const string & *file_name*)

Creates a new object of the class.

Parameters

file_name	complete file name of the configuration file to parse

4.4.2.2 FileConfigReader::~FileConfigReader ( )

Class destructor.

4.4.3 Member Function Documentation

4.4.3.1 vector< string > FileConfigReader::getFileList ( )

Returns the list of the input files to convert with full path.

Returns

the list of the input files to convert with full path

4.4.3.2 int FileConfigReader::getInputFormat ( )

Returns the format of the input files (Lecroy, Agilent, txt).

Returns

the format of the input files (Lecroy, Agilent, txt)

4.4.3.3 string FileConfigReader::getOutputName ( )

Returns the name of the output file to save.

Returns

the name of the output file to save

4.4.3.4 string FileConfigReader::getOutputPath ( )

Returns the location where the output files will be saved.

Returns

the location where the output files will be saved

4.4.3.5 UINTN FileConfigReader::getOutputTraceLength ( )

Returns the length of the trace to save.

Returns

the length of the trace to save in number of samples

4.4.3.6 UINTN FileConfigReader::getOutputTraceOffset ( )

Returns the number of samples to ignore during the conversion.

Returns

the number of samples to ignore during the conversion

4.4.3.7 UINTN FileConfigReader::getOutputTracePerFile ( )

Returns the number of traces to save in any output file.

Returns

the number of traces to save in any output file

4.4.3.8 template<typename ValueType > ValueType FileConfigReader::getValueOfKey (const string & *key*, ValueType const & *defaultValue* = `ValueType()`) const

Returns the value of a parsed key converted in ValueType.

Parameters

key	name of the key
ValueType	data type of the value of the key

Returns

the value of a parsed key converted in ValueType this function uses a template to get the value of the key converted in the desidered type.
used type: unsigned int, string.

4.4.3.9 bool FileConfigReader::keyExists (const string & *key*) const

Checks if a key exists in the list of parsed keys.

Parameters

key	name of the key

Returns

true if the key was parsed, false otherwise

The documentation for this class was generated from the following files:

- FileConfigReader.h
- FileConfigReader.cpp

4.5 FileReader Class Reference

This class parses Lecroy or Agilent files.

`#include <FileReader.h>`

Public Member Functions

- FileReader (FileConfigReader &settings)
  Creates a new object of the class.
- ~FileReader ()
  Class destructor.
- UINTN getNSample ()
  Returns the number of samples saved for each trace.
- char getFormatType ()
  Returns the data format type.
- UINTN getFormatSize ()
  Returns the size in bytes of the data format type.
- vector< uint8_t * > getTraces ()
  Returns the list of all the traces contained in the parsed files.

4.5.1 Detailed Description

This class parses Lecroy or Agilent files.

This class parses a list of files (Lecroy or Agilent), save the most important informations and then saves the traces of each file in a single vector. This vector is used to create the output files.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 FileReader::FileReader (FileConfigReader & *settings*)

Creates a new object of the class.

Parameters

settings	the file with the settings used to parse Lecroy/Agilent files.

4.5.2.2 FileReader::~FileReader ( )

Class destructor.

4.5.3 Member Function Documentation

4.5.3.1 UINTN FileReader::getFormatSize ( )

Returns the size in bytes of the data format type.

Returns

the size in byte of the data format type

4.5.3.2 char FileReader::getFormatType ( )

Returns the data format type.

Returns

the data format type several format types are available:

- for Lecroy files: 'b' for int8 and 'c' for int16
- for Agilent files: 'f' for float and 'd' for double

4.5.3.3 UINTN FileReader::getNSample ( )

Returns the number of samples saved for each trace.

Returns

the number of samples saved for each trace

4.5.3.4 vector< uint8_t * > FileReader::getTraces ( )

Returns the list of all the traces contained in the parsed files.

Returns

the list of all the traces contained in the parsed files

The documentation for this class was generated from the following files:

- FileReader.h
- FileReader.cpp

4.6 Utils Class Reference

This class provides some helpful static functions.

`#include <Utils.h>`

Static Public Member Functions

- static bool isHexText (string str)

  Returns true if the string is in hexadecimal format, false otherwise.
- static void HexToBin (string str, uint8_t *bin)

  Transforms a string in hexadecimal format into a binary format.
- static string CleanString (string str)

  Cleans a string.
- static string createOutputName (string name, unsigned int num, unsigned int pad)

  Creates a name with a pad of digit and a number at the end.
- static string adjustPath (string path)

  Check if a string terminates with '/', otherwise it adds this character.

4.6.1 Detailed Description

This class provides some helpful static functions.

4.6.2 Member Function Documentation

4.6.2.1 string Utils::adjustPath (string *path*) [static]

Check if a string terminates with '/', otherwise it adds this character.

Parameters

path	string to check

Returns

the same string terminated with '/'

4.6.2.2 string Utils::CleanString (string *str*) [static]

Cleans a string.

Parameters

str	string to clean

Returns

a string without space, '
', '', ''

4.6.2.3 string Utils::createOutputName (string *name,* unsigned int *num,* unsigned int *pad*) [static]

Creates a name with a pad of digit and a number at the end.

Parameters

name	name to modify
num	number to add at the end
pad	the number of digit to add (filled with zeros)

Returns

the name modified (name + zeros padding + number) this function is useful to create names in sequence.
for example: out0001, out0002, etc...

4.6.2.4 void Utils::HexToBin (string *str,* uint8_t * *bin*) [static]

Transforms a string in hexadecimal format into a binary format.

Parameters

str	string to convert
bin	result of the conversion

4.6.2.5 bool Utils::isHexText (string *str*) [static]

Returns true if the string is in hexadecimal format, false otherwise.

Parameters

str	string to check

Returns

true if the string is in hexadecimal format, false otherwise

The documentation for this class was generated from the following files:

- Utils.h
- Utils.cpp

Chapter 5

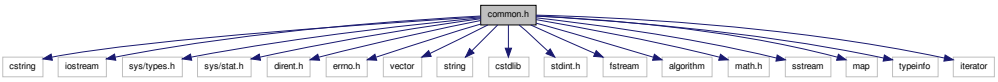
File Documentation

5.1 common.h File Reference

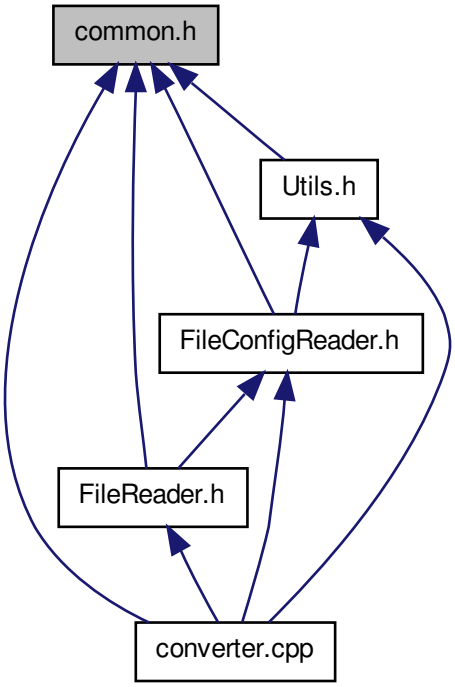
This file contains some commonly used includes, constants, defines, macros and typedefs.

```
#include <cstring>
#include <iostream>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <errno.h>
#include <vector>
#include <string>
#include <cstdlib>
#include <stdint.h>
#include <fstream>
#include <algorithm>
#include <math.h>
#include <sstream>
#include <map>
#include <typeinfo>
#include <iterator>
```

Include dependency graph for common.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct AgilentFileHeaderTag

  The type used to represent the header of an Agilent file.
- struct AgilentWaveformHeaderTag

  The type used to represent the header of each waveform in an Agilent file.
- struct AgilentWaveformDataHeaderTag

  The type used to represent the data header of each waveform in an Agilent file.

Defines

- #define **FILE_TYPE_LECROY** 1
- #define **FILE_TYPE_AGILENT** 2
- #define **FILE_TYPE_TXT** 3
- #define **DATE_TIME_STRING_LENGTH** 16
- #define **FRAME_STRING_LENGTH** 24
- #define **SIGNAL_STRING_LENGTH** 16

Typedefs

- typedef unsigned int UINTN

  The type used to represent unsigned integer number. This value should be equal to 2^n.
- typedef struct AgilentFileHeaderTag AgilentFileHeader

  The type used to represent the header of an Agilent file.
- typedef struct AgilentWaveformHeaderTag AgilentWaveformHeader

  The type used to represent the header of each waveform in an Agilent file.
- typedef struct AgilentWaveformDataHeaderTag AgilentWaveformDataHeader

  The type used to represent the data header of each waveform in an Agilent file.

5.1.1 Detailed Description

This file contains some commonly used includes, constants, defines, macros and typedefs. This file should be included by every .h file in the project.

User should not change this file directly.

The use of std namespace is declared.

5.1.2 Typedef Documentation

5.1.2.1 typedef struct AgilentFileHeaderTag AgilentFileHeader

The type used to represent the header of an Agilent file.

The struct represents the equivalent header of an Agilent file

5.1.2.2 typedef struct AgilentWaveformDataHeaderTag AgilentWaveformDataHeader

The type used to represent the data header of each waveform in an Agilent file.

The struct represents the equivalent data header of a waveform in an Agilent file

5.1.2.3 typedef struct AgilentWaveformHeaderTag AgilentWaveformHeader

The type used to represent the header of each waveform in an Agilent file.

The struct represents the equivalent header of a waveform in an Agilent file

5.1.2.4 typedef unsigned int UINTN

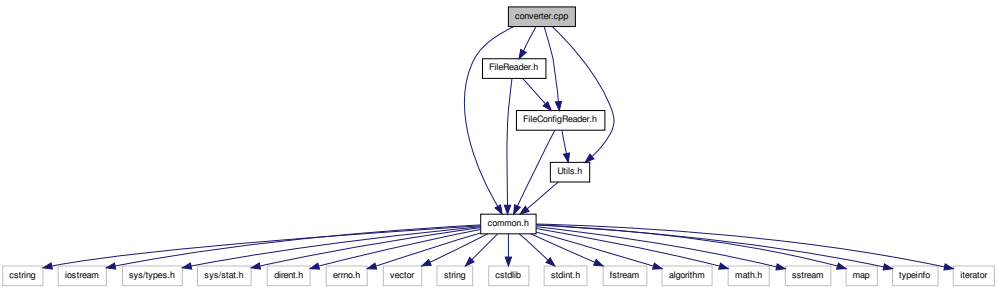
The type used to represent unsigned integer number. This value should be equal to 2^n.

5.2 converter.cpp File Reference

This is the file containing the main of the program.

```
#include "common.h"
#include "FileReader.h"
#include "FileConfigReader.h"
#include "Utils.h"
```

Include dependency graph for converter.cpp:



Functions

- void help ()

  Prints help informations.
- int main (int argc, char *argv[])

  Main function of the program.

5.2.1 Detailed Description

This is the file containing the main of the program.

5.2.2 Function Documentation

5.2.2.1 void help ( )

Prints help informations.

5.2.2.2 int main (int *argc*, char * *argv[]*)

Main function of the program.

Parameters

argc	
argv	

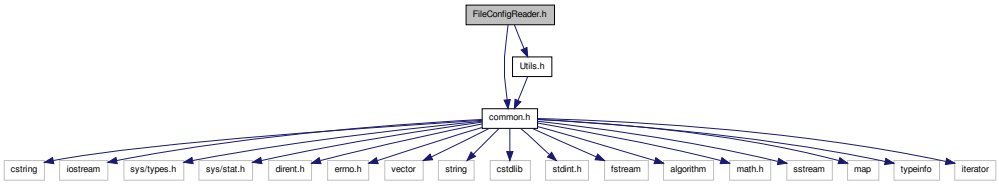
Returns

0 if everything went fine

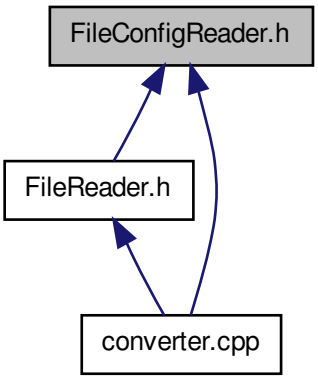
5.3 FileConfigReader.h File Reference

```
#include "common.h"
#include "Utils.h"
```

Include dependency graph for FileConfigReader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FileConfigReader

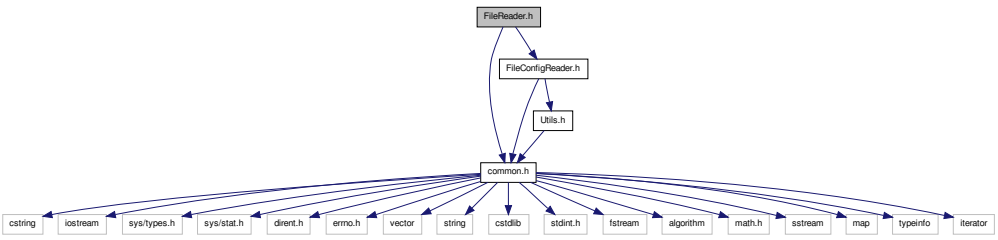
  This class parses a converter configuration file.

5.3.1 Detailed Description

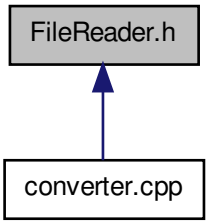
5.4 FileReader.h File Reference

```
#include "common.h"
#include "FileConfigReader.h"
```

Include dependency graph for FileReader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FileReader

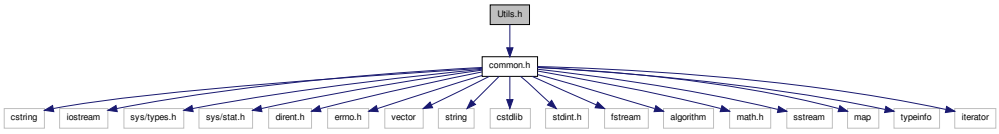
  This class parses Lecroy or Agilent files.

5.4.1 Detailed Description

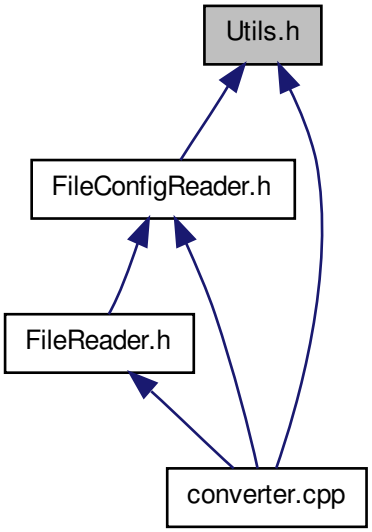
5.5 Utils.h File Reference

```
#include "common.h"
```

Include dependency graph for Utils.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Utils

  This class provides some helpful static functions.

5.5.1 Detailed Description

Index