Lib CEA Archive

1.0

Generated by Doxygen 1.8.2

Sun Dec 30 2012 10:02:09

Contents

1 Module Index

1.1 Modules

Here is a list of all modules:

Debug messages 2

2 Class Index

2.1 Class List

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

Config
Class for reading configuration structures from archives 2

Data
Class for reading data from archive 6

Header
Class for reading header from archive 13

ReadWriteArchive
The ReadWriteArchive class 14

3 File Index

3.1 File List

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

Config.cpp ??

Config.h ??

Data.cpp 19

Data.h ??

Header.cpp ??

Header.h ??

ReadWriteArchive.cpp 20

4 Module Documentation

4.1 Debug messages

Macros

- #define OFF 0

  Debug messages off.
- #define LEVEL1 1

  Show basic debug messages.
- #define LEVEL2 2

  Show verbose debug messages.

Variables

- int LIBARCHIVE_DEBUG

  Level of debug messages.

4.1.1 Detailed Description

4.1.2 Variable Documentation

4.1.2.1 int LIBARCHIVE_DEBUG

Level of debug messages.

Which level of debug messages should be displayed.

Definition at line 3 of file swap.cpp.

5 Class Documentation

5.1 Config Class Reference

Class for reading configuration structures from archives.

`#include <Config.h>`

Public Member Functions

- int read (FILE ∗fr, config ∗cfgGlobal)

  Reads configuration from archive.
- int write (FILE ∗fr, config ∗cfgGlobal)

  Writes configuration to archive.

Private Member Functions

- int writeSMPArcConfig (FILE ∗fr, int len, SmpArcConfig ∗aC)

  Writes Arc configuration to archive.
- int readSMPArcConfig (FILE ∗fr, int len, SmpArcConfig ∗aC)

  Reads Arc configuration from archive.
- int readSMPInstallConfig (FILE ∗fr, int len, SmpInstallConfig ∗iC)

  Reads Install configuration from archive.
- int writeSMPInstallConfig (FILE ∗fr, int len, SmpInstallConfig ∗iC)

  Writes Install configuration to archive.
- int readSMPConfig (FILE ∗fr, int len, SmpConfig ∗conf)

  Reads Main configuration from archive.
- int writeSMPConfig (FILE ∗fr, int len, SmpConfig ∗conf)

  Writes Main configuration from archive.
- int readSMPPqSettings (FILE ∗fr, int len, SmpPqSettings ∗pqS)

  Reads PQ configuration from archive.

5.1.1 Detailed Description

Class for reading configuration structures from archives.

In each archive is stored configuration before data. This class reads whole configuration and stores them to structure.

Definition at line 11 of file Config.h.

5.1.2 Member Function Documentation

5.1.2.1 int Config::read (FILE ∗ *fr*, config ∗ *cfgGlobal*)

Reads configuration from archive.

Begin read on actual position in file *fr*. Configuration is stored in *cfgGlobal*.

Parameters

fr	pointer to opened file.
cfgGlobal	pointer to allocated config structure where is configuration stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were realy read.

Definition at line 11 of file Config.cpp.

5.1.2.2 int Config::readSMPArcConfig (FILE ∗ *fr*, int *len*, SmpArcConfig ∗ *aC*) `[private]`

Reads Arc configuration from archive.

Begin read on actual position in file *fr*. Reads *len* bytes. Configuration is stored in *aC*.

Parameters

fr	pointer to opened file.
len	number of bytes which should be read.
aC	pointer to allocated SmpArcConfig structure where is configuration stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were realy read.

See Also

readSMPInstallConfig(), readSMPConfig(), readSMPPqSettings()

Definition at line 136 of file Config.cpp.

5.1.2.3 int Config::readSMPConfig (FILE ∗ *fr,* int *len,* SmpConfig ∗ *conf*) `[private]`

Reads Main configuration from archive.

Begin read on actual position in file *fr*. Reads *len* bytes. Configuration is stored in *conf*.

Parameters

fr	pointer to opened file.
len	number of bytes which should be read.
conf	pointer to allocated SmpInstallConfig structure where is configuration stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were realy read.

See Also

readSMPArcConfig(), readSMPInstallConfig(), readSMPPqSettings()

Definition at line 355 of file Config.cpp.

5.1.2.4 int Config::readSMPInstallConfig (FILE ∗ *fr,* int *len,* SmpInstallConfig ∗ *iC*) `[private]`

Reads Install configuration from archive.

Begin read on actual position in file *fr*. Reads *len* bytes. Configuration is stored in *iC*.

Parameters

fr	pointer to opened file.
len	number of bytes which should be read.
iC	pointer to allocated SmpInstallConfig structure where is configuration stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were realy read.

See Also

readSMPArcConfig(), readSMPConfig(), readSMPPqSettings()

Definition at line 226 of file Config.cpp.

5.1.2.5 int Config::readSMPPqSettings (FILE ∗ *fr,* int *len,* SmpPqSettings ∗ *pqS*) `[private]`

Reads PQ configuration from archive.

Begin read on actual position in file *fr*. Reads *len* bytes. Configuration is stored in *pqS*.

Parameters

fr	pointer to opened file.
len	number of bytes which should be read.
pqS	pointer to allocated SmpInstallConfig structure where is configuration stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were realy read.

See Also

readSMPArcConfig(), readSMPInstallConfig(), readSMPConfig()

Definition at line 486 of file Config.cpp.

5.1.2.6 int Config::write (FILE ∗ *fr*, config ∗ *cfgGlobal*)

Writes configuration to archive.

Begin write on actual position in file *fr*. Configuration is read from *cfgGlobal*.

Parameters

fr	pointer to opened file.
cfgGlobal	pointer to config structure with configuration.

Returns

0 if everything worked fine or difference between bytes which should be read and which were realy read.

Definition at line 67 of file Config.cpp.

5.1.2.7 int Config::writeSMPArcConfig (FILE ∗ *fr*, int *len*, SmpArcConfig ∗ *aC*) `[private]`

Writes Arc configuration to archive.

Begin write on actual position in file *fr*. Writes *len* bytes.

Parameters

fr	pointer to opened file.
len	number of bytes which should be written.
aC	pointer to SmpArcConfig structure with configuration.

Returns

0 if everything worked fine or difference between bytes which should be written and which were realy wrote.

See Also

readSMPInstallConfig(), readSMPConfig(), readSMPPqSettings()

Definition at line 180 of file Config.cpp.

5.1.2.8 int Config::writeSMPConfig (FILE ∗ *fr*, int *len*, SmpConfig ∗ *conf*) `[private]`

Writes Main configuration from archive.

Begin write on actual position in file *fr*. Writes *len* bytes.

Parameters

fr	pointer to opened file.
len	number of bytes which should be written.
conf	pointer to allocated SmpInstallConfig structure with data.

Returns

0 if everything worked fine or difference between bytes which should be written and which were realy wrote.

See Also

readSMPArcConfig(), readSMPInstallConfig(), readSMPPqSettings()

Definition at line 415 of file Config.cpp.

5.1.2.9 int Config::writeSMPInstallConfig (FILE ∗ *fr,* int *len,* SmpInstallConfig ∗ *iC*) [private]

Writes Install configuration to archive.

Begin write on actual position in file *fr*. Writes *len* bytes.

Parameters

fr	pointer to opened file.
len	number of bytes which should be written.
iC	pointer SmpInstallConfig structure with configuration.

Returns

0 if everything worked fine or difference between bytes which should be written and which were realy wrote.

See Also

readSMPArcConfig(), readSMPConfig(), readSMPPqSettings()

Definition at line 284 of file Config.cpp.

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

- Config.h
- Config.cpp

5.2 Data Class Reference

Class for reading data from archive.

```
#include <Data.h>
```

Public Member Functions

- Data (config ∗cfg)

  A constructor.
- virtual ∼Data ()

  A destructor does nothing.
- int readElmer (FILE ∗fr, int num, ElmerData ∗data)

  Reads electricity meter data from archive.

- int writeElmer (FILE ∗fr, int num, ElmerData ∗data)

  Reads electricity meter data from archive.
- int readMain (FILE ∗fr, int num, mainData ∗data)

  Reads main meter data from archive.

Private Member Functions

- int readMainTHDDB (FILE ∗fr, u8 configByte, float ∗t)

  Reads THD data from main archive.
- int readMainPDB (FILE ∗fr, u8 configByte, float ∗p)

  Reads power data from main archive.
- int readMainPDB (FILE ∗fr, u8 configByte, float ∗p, float ∗pm)

  Reads power data from main archive.
- int readMainfDB (FILE ∗fr, float ∗f)

  Reads frequency data from main archive.
- int readMainFlickerDB (FILE ∗fr, u8 configByte, u16 ∗fl)

  Reads flicker data from main archive.
- int readMainIDB (FILE ∗fr, u8 configByte, float ∗i)

  Reads current data from main archive.
- int readMainUDB (FILE ∗fr, u8 configByte, float ∗u)

  Reads voltage data from main archive.
- int readMainFiDB (FILE ∗fr, float ∗fi)

  Reads degree data from main archive.
- int readMainHarm (FILE ∗fr, config ∗cfg, harmData ∗hData)

  Reads harmonics data from main archive.
- int readHarmOnePhase (FILE ∗fr, float ∗hData, int len, float mul, bool even, bool sign)

  Reads harmonics data for one phase.
- int decodeMTP (u16 inMTP)

  Decode MTP.
- int decodeMTN (u16 inMTN)

  Decode MTN.

Private Attributes

- config ∗ cfg

  configuration for current archive.
- u16 **MTN**
- u16 **MTNN**
- u16 **MTP**
- u16 **MTPN**

5.2.1 Detailed Description

Class for reading data from archive.

Definition at line 8 of file Data.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Data::Data (config ∗ *cfg*)

A constructor.

Stores configuration from *cfg* to private variable cfg. Decode MTP and MTP from *cfg* and stores them to private varibales.

Parameters

cfg	structure with configuration.

Definition at line 6 of file Data.cpp.

5.2.3 Member Function Documentation

5.2.3.1 int Data::decodeMTN (u16 *inMTN*) `[private]`

Decode MTN.

Decodes MTN from its encoded form stored in archive.

Parameters

inMTN	encoded MTN value.

Returns

decoded MTN value.

Definition at line 715 of file Data.cpp.

5.2.3.2 int Data::decodeMTP (u16 *inMTP*) `[private]`

Decode MTP.

Decodes MTP from its encoded form stored in archive.

Parameters

inMTP	encoded MTP value.

Returns

decoded MTP value.

Definition at line 708 of file Data.cpp.

5.2.3.3 int Data::readElmer (FILE ∗ *fr,* int *num,* ElmerData ∗ *data*)

Reads electricity meter data from archive.

Begin read on actual position in file *fr*. Reads *num* data cycles. Data are stored in *data*.

Parameters

fr	pointer to opened file.
num	number of data cycles stored in archive.
data	pointer to allocated electricity meter data structure where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

Definition at line 255 of file Data.cpp.

5.2.3.4 int Data::readHarmOnePhase (FILE ∗ *fr,* float ∗ *hData,* int *len,* float *mul,* bool *even,* bool *sign*) [private]

Reads harmonics data for one phase.

Begin read on actual position in file *fr*. Data are stored in *hData*.

Parameters

fr	pointer to opened file.
hData	pointer to allocated harmData structure where data are stored.
len	number of harmonics stored per phase.
mul	multiplier with which are read data multiplied.
even	even or odd.
sign	if store data are unsigne or signed 16 bit value.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainHarm()

Definition at line 622 of file Data.cpp.

5.2.3.5 int Data::readMain (FILE ∗ *fr,* int *num,* mainData ∗ *data*)

Reads main meter data from archive.

Begin read on actual position in file *fr*. Reads *num* data cycles. Data are stored in *data*.

Parameters

fr	pointer to opened file.
num	number of data cycles stroed in archive.
data	pointer to allocated electricity meter data structure where data are stored.

Returns

0 if everything worked fine, -1 on error or difference between bytes which should be read and which were really read.

Definition at line 16 of file Data.cpp.

5.2.3.6 int Data::readMainfDB (FILE ∗ *fr,* float ∗ *f*) [private]

Reads frequency data from main archive.

Begin read on actual position in file *fr*. Data are stored in *f*.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
f	pointer to float array where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainPDB(), readMainFlickerDB(), readMainFlickerDB(), readMainIDB(), readMainUD-B(), readMainFiDB(), readMainHarm()

Definition at line 530 of file Data.cpp.

5.2.3.7 int Data::readMainFiDB (FILE ∗ *fr,* float ∗ *fi*) [private]

Reads degree data from main archive.

Begin read on actual position in file *fr*. Data are stored in *fi*.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
fi	pointer to float array where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainPDB(), readMainfDB(), readMainFlickerDB(), readMainFlickerDB(), readMainID-B(), readMainUDB(), readMainHarm()

Definition at line 606 of file Data.cpp.

5.2.3.8 int Data::readMainFlickerDB (FILE ∗ *fr,* u8 *configByte,* u16 ∗ *fl*) [private]

Reads flicker data from main archive.

Begin read on actual position in file *fr*. Data are stored in *fl*.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
fl	pointer to float array where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainPDB(), readMainfDB(), readMainFlickerDB(), readMainIDB(), readMainUDB(), readMainFiDB(), readMainHarm()

Definition at line 551 of file Data.cpp.

5.2.3.9 int Data::readMainHarm (FILE ∗ *fr,* config ∗ *cfg,* harmData ∗ *hData*) [private]

Reads harmonics data from main archive.

Begin read on actual position in file *fr*. Data are stored in *hData*.

Parameters

fr	pointer to opened file.
cfg	structure with configuration.
hData	pointer to allocated harmData structure where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainPDB(), readMainfDB(), readMainFlickerDB(), readMainFlickerDB(), readMainID-B(), readMainUDB(), readMainFiDB()

Definition at line 666 of file Data.cpp.

5.2.3.10 int Data::readMainIDB (FILE ∗ *fr*, u8 *configByte*, float ∗ *i*) [private]

Reads current data from main archive.

Begin read on actual position in file *fr*. Data are stored in *i*.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
i	pointer to float array where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainPDB(), readMainfDB(), readMainFlickerDB(), readMainFlickerDB(), readMainUD-B(), readMainFiDB(), readMainHarm()

Definition at line 507 of file Data.cpp.

5.2.3.11 int Data::readMainPDB (FILE ∗ *fr*, u8 *configByte*, float ∗ *p*) [private]

Reads power data from main archive.

Begin read on actual position in file *fr*. Data are stored in *p*. Used when ony positive power can be read.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
p	pointer to float array where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainfDB(), readMainFlickerDB(), readMainFlickerDB(), readMainIDB(), readMainUD-B(), readMainFiDB(), readMainHarm()

Definition at line 447 of file Data.cpp.

5.2.3.12 int Data::readMainPDB (FILE ∗ *fr*, u8 *configByte*, float ∗ *p*, float ∗ *pm*) [private]

Reads power data from main archive.

Begin read on actual position in file *fr*. Data are stored in *p*. Used when positive and negative power can be read.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
p	pointer to float array where positive data are stored.
p	pointer to float array where negative data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainfDB(), readMainFlickerDB(), readMainFlickerDB(), readMainIDB(), readMainUD-B(), readMainFiDB(), readMainHarm()

Definition at line 469 of file Data.cpp.

5.2.3.13 int Data::readMainTHDDB (FILE ∗ *fr*, u8 *configByte*, float ∗ *t*) [private]

Reads THD data from main archive.

Begin read on actual position in file *fr*. Data are stored in *t*.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
t	pointer to float array where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainPDB(), readMainfDB(), readMainFlickerDB(), readMainFlickerDB(), readMainIDB(), readMainUDB(), readMainFiDB(), readMainHarm()

Definition at line 424 of file Data.cpp.

5.2.3.14 int Data::readMainUDB (FILE ∗ *fr*, u8 *configByte*, float ∗ *u*) [private]

Reads voltage data from main archive.

Begin read on actual position in file *fr*. Data are stored in *u*.

Parameters

fr	pointer to opened file.
configByte	configuration byte with number of phases which should be read.
u	pointer to float array where data are stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were really read.

See Also

readMainTHDDB(), readMainPDB(), readMainfDB(), readMainFlickerDB(), readMainFlickerDB(), readMainID-B(), readMainFiDB(), readMainHarm()

Definition at line 581 of file Data.cpp.

5.2.3.15 int Data::writeElmer (FILE ∗ *fr*, int *num*, ElmerData ∗ *data*)

Reads electricity meter data from archive.

Begin write on actual position in file *fr*. Writes *num* data cycles.

Parameters

fr	pointer to opened file.
num	number of data cycles which should be stored in archive.
data	pointerelectricity meter data structure.

Returns

0 if everything worked fine or difference between bytes which should be written and which were really wrote.

Definition at line 342 of file Data.cpp.

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

- Data.h
- Data.cpp

5.3 Header Class Reference

Class for reading header from archive.

```
#include <Header.h>
```

Public Member Functions

- int read (FILE ∗fr, headerInfo ∗info)

  Reads header from archive.
- int write (FILE ∗fr, headerInfo ∗info)

  Writes header from archive.

5.3.1 Detailed Description

Class for reading header from archive.

At the beginning of each archive is header which containst basic information about archive.

Definition at line 11 of file Header.h.

5.3.2 Member Function Documentation

5.3.2.1 int Header::read (FILE ∗ *fr*, headerInfo ∗ *info*)

Reads header from archive.

Begin read on actual position in file *fr* which should be begin of the file. Header is stored in *info*.

Parameters

fr	pointer to opened file.
info	pointer to allocated header structure where are data stored.

Returns

0 if everything worked fine or difference between bytes which should be read and which were realy read.

Definition at line 13 of file Header.cpp.

5.3.2.2 int Header::write (FILE ∗ *fr*, headerInfo ∗ *info*)

Writes header from archive.

Begin write on actual position in file *fr* which should be begin of the file.

Parameters

fr	pointer to opened file.
info	pointer to header structure with data.

Returns

0 if everything worked fine or difference between bytes which should be write and which were realy wrote.

Definition at line 34 of file Header.cpp.

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

- Header.h
- Header.cpp

5.4 ReadWriteArchive Class Reference

The ReadWriteArchive class.

```
#include <ReadWriteArchive.h>
```

Public Member Functions

- ReadWriteArchive (const char ∗archive)

  A constructor.
- virtual ∼ReadWriteArchive ()

  A destructor.
- int readMainArchive (mainData ∗∗data, config ∗∗cfg, int ∗nRec, int dev)

  Read Main Archive.
- int readElmerArchive (ElmerData ∗∗data, int ∗nRec)

  Read Main Archive.
- int freeMainData (mainData ∗∗data)

Free Main Data.
- int openArchive ()

  Open archive.
- int closeArchive ()

  Close archive.
- void setDebug (int debugLevel)

  Set Debug Level.
- int getNDevices ()

  Get Number of Devices.
- void setDevice (int num)

  Set Device number.

Private Member Functions

- int selectMainArchive (char ∗∗mainArchives=NULL)

  Select Main Archive.
- int allocateMainData (int nArch, config ∗cfg, mainData ∗∗data)

  Allocate Main Data structure.
- int getHarmNum (config ∗cfg)

  Get Harmonics Number.
- int getHarmLength (config ∗cfg)

  Get Harmonics Length.

Private Attributes

- char ∗ archive

  Path to archive.
- char ∗ tmpDir

  Path to the temprorary dir where archive files will be unzipped.
- int nDev

  Number of devices stored in current archive.
- int nRec

  Number of write cycles stored in current archive.
- int curDev

  Selected device.

5.4.1 Detailed Description

The ReadWriteArchive class.

This is the main class of whole library. Public members from here are API for library.

Examples:

example.cpp.

Definition at line 38 of file ReadWriteArchive.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ReadWriteArchive::ReadWriteArchive (const char ∗ *archive*)

A constructor.

Stores *archive* path to private variable. Allocates memory for variables and initializes random generator.

Parameters

archive	path to archive which shloul be read.

Definition at line 15 of file ReadWriteArchive.cpp.

5.4.2.2 ReadWriteArchive::∼ReadWriteArchive ( ) `[virtual]`

A destructor.

Frees variables.

Definition at line 31 of file ReadWriteArchive.cpp.

5.4.3 Member Function Documentation

5.4.3.1 int ReadWriteArchive::allocateMainData (int *nArch,* config ∗ *cfg,* mainData ∗∗ *data*) `[private]`

Allocate Main Data structure.

Allocates memory for mainData structure

Parameters

nArch	number of read cycle stored in archive
cfg	pointer to structure with configuration to current archive
data	pointer to data structure to allocate

Returns

0 if everything worked fine or negative value on error.

Definition at line 335 of file ReadWriteArchive.cpp.

5.4.3.2 int ReadWriteArchive::closeArchive ( )

Close archive.

Closes archive on path passed on in constructor.

Returns

0 if everything worked fine or negative value on error.

Examples:

example.cpp.

Definition at line 312 of file ReadWriteArchive.cpp.

5.4.3.3 int ReadWriteArchive::freeMainData (mainData ∗∗ *data*)

Free Main Data.

Free mainData structure allocated in readMainArchive.

Parameters

data	data structure to free

Returns

0 if everything worked fine or negative value on error.

Examples:

example.cpp.

Definition at line 450 of file ReadWriteArchive.cpp.

5.4.3.4 int ReadWriteArchive::getHarmLength (config ∗ *cfg*) `[private]`

Get Harmonics Length.

Counts number of bytes taken by harmonics.

Parameters

cfg	pointer to structure with configuration to current archive

Returns

Number of bytes taken by harmonics.

Definition at line 494 of file ReadWriteArchive.cpp.

5.4.3.5 int ReadWriteArchive::getHarmNum (config ∗ *cfg*) `[private]`

Get Harmonics Number.

Counts number of harmonics stored in archive.

Parameters

cfg	pointer to structure with configuration to current archive

Returns

Number of harmonics per phase

Definition at line 487 of file ReadWriteArchive.cpp.

5.4.3.6 int ReadWriteArchive::getNDevices ( )

Get Number of Devices.

Return number of devices stored in current archive - private variable *nDev*.

Returns

number of devices - private variable *nDev*.

Definition at line 151 of file ReadWriteArchive.cpp.

5.4.3.7 int ReadWriteArchive::openArchive ( )

Open archive.

Opens archive on path passed on in constructor.

Returns

0 if everything worked fine or negative value on error.

Examples:

example.cpp.

Definition at line 209 of file ReadWriteArchive.cpp.

5.4.3.8 int ReadWriteArchive::readElmerArchive (ElmerData ∗∗ *data*, int ∗ *nRec*)

Read Main Archive.

∗Not yet implemented.

Definition at line 148 of file ReadWriteArchive.cpp.

5.4.3.9 int ReadWriteArchive::readMainArchive (mainData ∗∗ *data*, config ∗∗ *cfg*, int ∗ *nRec*, int *dev*)

Read Main Archive.

∗From opened archive reads configuration, selects achive file with main data and reads it.

Parameters

data	unalocated data structure.
cfg	unalocated config structure.
dev	number of device file from which data schould be read.
nRec	number of recors - empty, will be stored there.

Returns

0 if everything worked fine or negative value on error.

Examples:

example.cpp.

Definition at line 36 of file ReadWriteArchive.cpp.

5.4.3.10 int ReadWriteArchive::selectMainArchive (char ∗∗ *mainArchives* = `NULL`) `[private]`

Select Main Archive.

Selects main archives file from unzipped archive files in temporary folder.

Parameters

manArchives	pointer array of temporary paths to unzipped main archive files

Returns

number of found archives or negative value on error.

Definition at line 156 of file ReadWriteArchive.cpp.

5.4.3.11 void ReadWriteArchive::setDebug (int *debugLevel*)

Set Debug Level.

Sets level of debug messages which shold be displayed.

Parameters

debugLevel	required level (OFF, LEVEL1, LEVEL2)

Examples:

example.cpp.

Definition at line 331 of file ReadWriteArchive.cpp.

5.4.3.12 void ReadWriteArchive::setDevice (int *num*)

Set Device number.

Sets number of device which shuld be used.

Parameters

device	number - private variable *nDev*.

Definition at line 520 of file ReadWriteArchive.cpp.

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

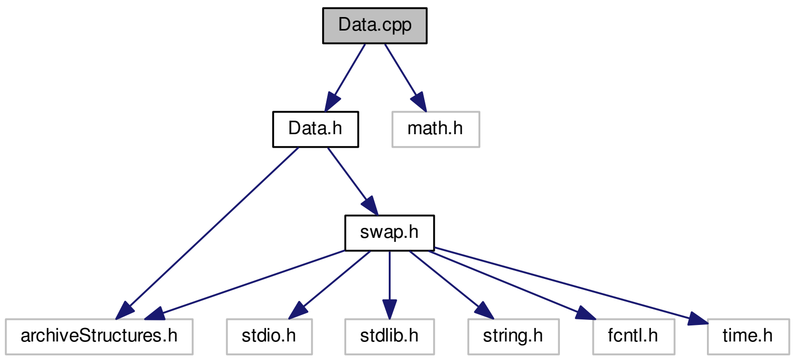
- ReadWriteArchive.h
- ReadWriteArchive.cpp

6 File Documentation

6.1 Data.cpp File Reference

```
#include "Data.h"
#include "math.h"
```

Include dependency graph for Data.cpp:



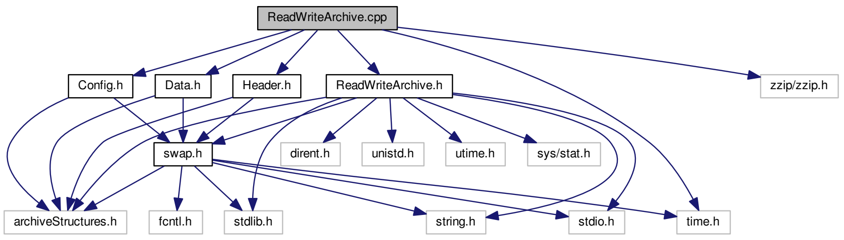
Macros

- #define **PI** 3.14159265358979323846264338327950288419716939937510

6.2 ReadWriteArchive.cpp File Reference

```
#include "ReadWriteArchive.h"
#include "Header.h"
#include "Config.h"
#include "Data.h"
#include <time.h>
#include <zzip/zzip.h>
```

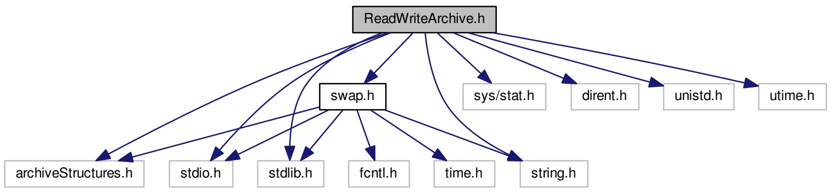
Include dependency graph for ReadWriteArchive.cpp:



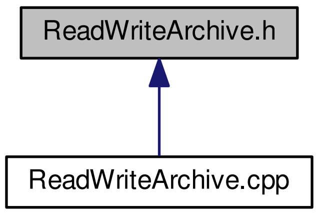
6.3 ReadWriteArchive.h File Reference

```
#include "archiveStructures.h"
#include "swap.h"
#include <sys/stat.h>
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <utime.h>
```

Include dependency graph for ReadWriteArchive.h:



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



Classes

- class ReadWriteArchive

  The ReadWriteArchive class.

Macros

- #define WRITEBUFFERSIZE (8192)
- #define **MAX_SAME_ARCHIVES** 10

6.3.1 Macro Definition Documentation

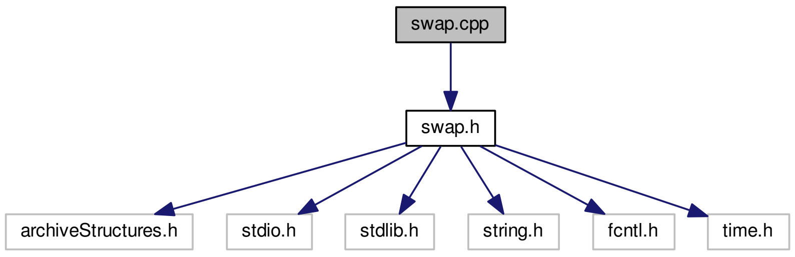
6.3.1.1 #define WRITEBUFFERSIZE (8192)

Library API

Definition at line 29 of file ReadWriteArchive.h.

6.4 swap.cpp File Reference

```
#include "swap.h"
```
Include dependency graph for swap.cpp:



Functions

- float swapFloat (const float in)

  Swap bytes in 4 byte float.

Variables

- int LIBARCHIVE_DEBUG = OFF

  Level of debug messages.

6.4.1 Function Documentation

6.4.1.1 float swapFloat (const float *in*)

Swap bytes in 4 byte float.

Swap bytes in 64 bit integer variable *x*.

See Also

bswap64(), bswap32(), bswap16()

Parameters

in	float which shloud be converted.

Returns

converted float.

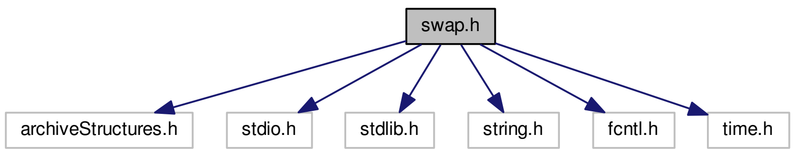
Definition at line 5 of file swap.cpp.

6.5 swap.h File Reference

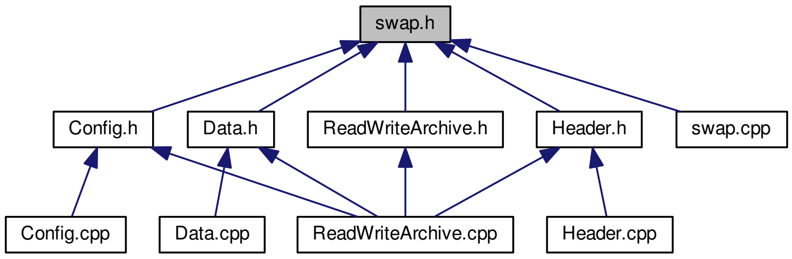
Macros and function for byte swap. Settings for displaying debug messages.

```
#include "archiveStructures.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <time.h>
```

Include dependency graph for swap.h:



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



Macros

- #define bswap16(x) ((x)>>8 | ((x)&255)<<8)
- #define bswap32(x)
- #define bswap64(x)
- #define OFF 0

    Debug messages off.
- #define LEVEL1 1

    Show basic debug messages.
- #define LEVEL2 2

    Show verbose debug messages.

Functions

- float swapFloat (const float in)

    Swap bytes in 4 byte float.

Variables

- int LIBARCHIVE_DEBUG

    Level of debug messages.

6.5.1 Detailed Description

Macros and function for byte swap. Settings for displaying debug messages. Some data are stored in little-endian format and we use big-endian machine. Theese functions are used for converting between little and big endian format. Global variable LIBARCHIVE_DEBUG which is used for storing level of debug messeges which should be shown.

Definition in file swap.h.

6.5.2 Macro Definition Documentation

6.5.2.1 #define bswap16(*x*) ((x)>>8 | ((x)&255)<<8)

See Also

bswap64(), bswap32(), swapFloat()

Swap bytes in 16 bit integer variable *x*.

Definition at line 25 of file swap.h.

6.5.2.2 #define bswap32(*x*)

Value:

```
((((x) & 0x000000ff) << 24) |       \
                         (((x) & 0x0000ff00) << 8)  |       \
                         (((x) & 0x00ff0000) >> 8)  |       \
                         (((x) & 0xff000000) >> 24))
```

See Also

bswap64(), bswap16(), swapFloat()

Swap bytes in 32 bit integer variable *x*.

Definition at line 33 of file swap.h.

6.5.2.3 #define bswap64(*x*)

Value:

```
( (x << 56) & 0xff00000000000000UL ) | \
                     ( (x << 40) & 0x00ff000000000000UL ) | \
                     ( (x << 24) & 0x0000ff0000000000UL ) | \
                     ( (x <<  8) & 0x000000ff00000000UL ) | \
                     ( (x >>  8) & 0x00000000ff000000UL ) | \
                     ( (x >> 24) & 0x0000000000ff0000UL ) | \
                     ( (x >> 40) & 0x000000000000ff00UL ) | \
                     ( (x >> 56) & 0x00000000000000ffUL )
```

See Also

bswap32(), bswap16(), swapFloat()

Swap bytes in 64 bit integer variable *x*.

Definition at line 44 of file swap.h.

6.5.3 Function Documentation

6.5.3.1 float swapFloat (const float *in*)

Swap bytes in 4 byte float.

Swap bytes in 64 bit integer variable *x*.

See Also

bswap64(), bswap32(), bswap16()

Parameters

in	float which shloud be converted.

Returns

converted float.

Definition at line 5 of file swap.cpp.

7 Example Documentation

7.1 example.cpp

This is an example of how to use the ReadWriteArchive class.

```
void main()
{
    mainData *mD;
    config *cfg;

    ReadWriteArchive* rwa = new ReadWriteArchive
      ("/path/Device.cea");
    rwa->setDebug(LEVEL1);
    rwa->openArchive();

    int result = rwa->readMainArchive(&mD,&cfg,&nRec);
    if(result == 0)
    {
    //display data somehow
    }

    if (result != -1 && result != -2)
    {
        rwa->freeMainData(nRec,&mD);
        free(cfg);
    }
    rwa->closeArchive();
    delete rwa;
}
```

Index