TITLE

AUTHOR

Version 1.0

CREATEDATE

# Table of Contents

Table of contents

# Todo List

##### Member [awaitingPattern\_mode](#AAAAAAAABV)

Need to be deleted.

##### Member [dataFile](#AAAAAAAABZ) [255]

Need to be deleted.

##### Member [dbg\_hex\_print](#AAAAAAAAGO) (BYTE \*buffer, size\_t len)

use this function in new debug print system

##### Member [deqUdp::readData](#AAAAAAAAIJ) ([ssBuffer](#AAAAAAAAES) \*addr, size\_t \*len, in\_addr \*ip\_from=0, bool peek=false)

not needed to realize all sendData check ability to use base method sendData of parent class

##### Member [deqUdp::sendData](#AAAAAAAAIL) ([ssBuffer](#AAAAAAAAES) \*buf)

not needed to realize all sendData check ability to use base method sendData of parent class

##### Member [equipListenPolling](#AAAAAAAAGD) (void \*user)

Listening equipment answer - status vector:

##### Member [equipListenPolling](#AAAAAAAAGG) (void \*)

add to [commonFuncsMgr](#AAAAAAAAGC) class as static method

##### Member [fileRead](#AAAAAAAAGN) (char \*fname, BYTE \*\*buffer, size\_t \*sz)

reorganize function to reading xml-files for future purposes

##### Member [if\_name](#AAAAAAAACA) [255]

Need to be deleted.

##### Member [listen\_mode](#AAAAAAAABU)

Need to be deleted.

##### Member [pattern\_found](#AAAAAAAABW)

Need to be deleted.

##### Member [patternFile](#AAAAAAAABX) [255]

Need to be deleted.

##### Member [process\_cmdLine](#AAAAAAAAGM) (int argc, char \*argv[])

reorganize process to external library

##### Member [reactionFile](#AAAAAAAABY) [255]

Need to be deleted.

##### Member [srvAppLayer::encodeBlock](#AAAAAAAAIW) (rcsCmd \*, BYTE \*\*)

nobody uses this method. need to be deleted.

##### Member [srvAppLayer::encodeFuncResult](#AAAAAAAAIY) (rcsCmd \*in\_cmd, rcsCmd \*out\_cmd)

Need a refactoring.

##### Member [srvAppLayer::equip\_read\_data](#AAAAAAAAJA) (BYTE \*, size\_t \*)

too strange method.

##### Member [srvAppLayer::equip\_reading\_event](#AAAAAAAAJC) ()

too strange method.

##### Member [srvAppLayer::execMessage](#AAAAAAAAJE) (rcsCmd \*ss\_cmd)

ServiceState vector need to change before function calling.

##### Member [srvAppLayer::Functions](#AAAAAAAAJG) [100]

array not the best data structure for this purposes.

# Directory Hierarchy

## Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

src pagenum

arg\_parser pagenum

buffer pagenum

deqUdp pagenum

functions pagenum

srvAppLayer pagenum

functionNode pagenum

# Class Index

## Class List

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

[**ap\_Option**](#AAAAAAAACQ) pagenum

[**ap\_Record**](#AAAAAAAACZ) pagenum

[**Arg\_parser**](#AAAAAAAACM) pagenum

[**buffer**](#AAAAAAAAEN) **(Simple queue of bytes )** pagenum

[**commonFuncsMgr**](#AAAAAAAAGC) **(Common functions manager implementation (set of function independs on target) )** pagenum

[**deqUdp**](#AAAAAAAAFX) **(Udp communications (based on udp\_port) with queues for listening and sending )** pagenum

[**functionNode**](#AAAAAAAAHI) **(Function node interface header )** pagenum

[**param\_desc**](#AAAAAAAAHN) **(Parameter description )** pagenum

[**serviceState**](#AAAAAAAAIA) **(**[**stateVector\_type**](#AAAAAAAAIB) **structural field )** pagenum

[**specFuncsMgr**](#AAAAAAAAGJ) **(Special functions set manager )** pagenum

[**srvAppLayer**](#AAAAAAAAGQ) **(Application core layer implementaion )** pagenum

[**ssBlock**](#AAAAAAAAEV) **(**[**ssBuffer**](#AAAAAAAAES) **list entry )** pagenum

[**ssBuffer**](#AAAAAAAAES) **(List (deque) implementaion for storing** [**ssBlock**](#AAAAAAAAEV) **elements )** pagenum

[**stateVector\_type**](#AAAAAAAAIB) **(Main vector of service base states )** pagenum

# File Index

## File List

Here is a list of all files with brief descriptions:

**src/**[**\_auto\_config.h**](#AAAAAAAAAB) pagenum

**src/**[**\_global.cpp**](#AAAAAAAABD) **(Global environment )** pagenum

**src/**[**\_global.h**](#AAAAAAAACH) **(Global environment interface header )** pagenum

**src/**[**auto\_config.h**](#AAAAAAAADK) pagenum

**src/**[**config.h**](#AAAAAAAAEY) pagenum

**src/**[**console\_out.cpp**](#AAAAAAAAEZ) **(Aided functions to process\_cmdLine )** pagenum

**src/**[**console\_out.h**](#AAAAAAAAFP) **(Aided functions interface header )** pagenum

**src/**[**main.cpp**](#AAAAAAAAGL) **(Programm entry point )** pagenum

**src/**[**SIG\_handler.cpp**](#AAAAAAAAHA) **(System signals handlers manager )** pagenum

**src/**[**SIG\_handler.h**](#AAAAAAAAHE) **(System signals handlers manager interface header )** pagenum

**src/arg\_parser/**[**carg\_parser.cpp**](#AAAAAAAACJ) pagenum

**src/arg\_parser/**[**carg\_parser.h**](#AAAAAAAACY) pagenum

**src/buffer/**[**buffer.cpp**](#AAAAAAAAEM) **(Class** [**buffer**](#AAAAAAAAEN) **implementation )** pagenum

**src/buffer/**[**buffer.h**](#AAAAAAAAEP) **(Class** [**buffer**](#AAAAAAAAEN) **interface header )** pagenum

**src/buffer/**[**ssBuffer.cpp**](#AAAAAAAAER) **(Class** [**ssBuffer**](#AAAAAAAAES) **implementation )** pagenum

**src/buffer/**[**ssBuffer.h**](#AAAAAAAAEU) **(Class** [**ssBuffer**](#AAAAAAAAES) **interface header )** pagenum

**src/deqUdp/**[**deqUdp.cpp**](#AAAAAAAAFW) **(Class** [**deqUdp**](#AAAAAAAAFX) **implementation )** pagenum

**src/deqUdp/**[**deqUdp.h**](#AAAAAAAAFZ) **(Class** [**deqUdp**](#AAAAAAAAFX) **interface header )** pagenum

**src/functions/**[**commonFuncsMgr.cpp**](#AAAAAAAAGB) **(Class** [**commonFuncsMgr**](#AAAAAAAAGC) **interface header )** pagenum

**src/functions/**[**commonFuncsMgr.h**](#AAAAAAAAGF) **(Class** [**commonFuncsMgr**](#AAAAAAAAGC) **interface header )** pagenum

**src/functions/**[**specFuncsMgr.h**](#AAAAAAAAGI) **(Class** [**specFuncsMgr**](#AAAAAAAAGJ) **interface header )** pagenum

**src/srvAppLayer/**[**srvAppLayer.cpp**](#AAAAAAAAHR) **(Class** [**srvAppLayer**](#AAAAAAAAGQ) **implementation )** pagenum

**src/srvAppLayer/**[**srvAppLayer.h**](#AAAAAAAAHZ) **(Class** [**srvAppLayer**](#AAAAAAAAGQ) **interface header )** pagenum

**src/srvAppLayer/functionNode/**[**functionNode.cpp**](#AAAAAAAAHH) **(Class** [**functionNode**](#AAAAAAAAHI) **implementation )** pagenum

**src/srvAppLayer/functionNode/**[**functionNode.h**](#AAAAAAAAHK) **(Class** [**functionNode**](#AAAAAAAAHI) **interface header )** pagenum

**src/srvAppLayer/functionNode/**[**param\_desc.cpp**](#AAAAAAAAHM) **(Class** [**param\_desc**](#AAAAAAAAHN) **implementation )** pagenum

**src/srvAppLayer/functionNode/**[**param\_desc.h**](#AAAAAAAAHP) **(Class** [**param\_desc**](#AAAAAAAAHN) **interface header )** pagenum

# Directory Documentation

## src/arg\_parser/ Directory Reference

Directory dependency graph for src/arg\_parser/:

### Files

file [carg\_parser.cpp](#AAAAAAAACJ)

file [carg\_parser.h](#AAAAAAAACY)

## src/buffer/ Directory Reference

Directory dependency graph for src/buffer/:

### Files

file [buffer.cpp](#AAAAAAAAEM)

*Class* [*buffer*](#AAAAAAAAEN) *implementation.*

file [buffer.h](#AAAAAAAAEP)

*Class* [*buffer*](#AAAAAAAAEN) *interface header.*

file [ssBuffer.cpp](#AAAAAAAAER)

*Class* [*ssBuffer*](#AAAAAAAAES) *implementation.*

file [ssBuffer.h](#AAAAAAAAEU)

*Class* [*ssBuffer*](#AAAAAAAAES) *interface header.*

## src/deqUdp/ Directory Reference

Directory dependency graph for src/deqUdp/:

### Files

file [deqUdp.cpp](#AAAAAAAAFW)

*Class* [*deqUdp*](#AAAAAAAAFX) *implementation.*

file [deqUdp.h](#AAAAAAAAFZ)

*Class* [*deqUdp*](#AAAAAAAAFX) *interface header.*

## src/srvAppLayer/functionNode/ Directory Reference

Directory dependency graph for src/srvAppLayer/functionNode/:

### Files

file [functionNode.cpp](#AAAAAAAAHH)

*Class* [*functionNode*](#AAAAAAAAHI) *implementation.*

file [functionNode.h](#AAAAAAAAHK)

*Class* [*functionNode*](#AAAAAAAAHI) *interface header.*

file [param\_desc.cpp](#AAAAAAAAHM)

*Class* [*param\_desc*](#AAAAAAAAHN) *implementation.*

file [param\_desc.h](#AAAAAAAAHP)

*Class* [*param\_desc*](#AAAAAAAAHN) *interface header.*

## src/functions/ Directory Reference

Directory dependency graph for src/functions/:

### Files

file [commonFuncsMgr.cpp](#AAAAAAAAGB)

*Class* [*commonFuncsMgr*](#AAAAAAAAGC) *interface header.*

file [commonFuncsMgr.h](#AAAAAAAAGF)

*Class* [*commonFuncsMgr*](#AAAAAAAAGC) *interface header.*

file [specFuncsMgr.h](#AAAAAAAAGI)

*Class* [*specFuncsMgr*](#AAAAAAAAGJ) *interface header.*

## src/ Directory Reference

Directory dependency graph for src/:

### Directories

directory [arg\_parser](#AAAAAAAAOF)

directory [buffer](#AAAAAAAAOG)

directory [deqUdp](#AAAAAAAAOH)

directory [functions](#AAAAAAAAOJ)

directory [srvAppLayer](#AAAAAAAAOL)

### Files

file [\_auto\_config.h](#AAAAAAAAAB)

file [\_global.cpp](#AAAAAAAABD)

*Global environment.*

file [\_global.h](#AAAAAAAACH)

*Global environment interface header.*

file [auto\_config.h](#AAAAAAAADK)

file [config.h](#AAAAAAAAEY)

file [console\_out.cpp](#AAAAAAAAEZ)

*aided functions to process\_cmdLine*

file [console\_out.h](#AAAAAAAAFP)

*aided functions interface header*

file [main.cpp](#AAAAAAAAGL)

*Programm entry point.*

file [SIG\_handler.cpp](#AAAAAAAAHA)

*System signals handlers manager.*

file [SIG\_handler.h](#AAAAAAAAHE)

*System signals handlers manager interface header.*

## src/srvAppLayer/ Directory Reference

Directory dependency graph for src/srvAppLayer/:

### Directories

directory [functionNode](#AAAAAAAAOI)

### Files

file [srvAppLayer.cpp](#AAAAAAAAHR)

*Class* [*srvAppLayer*](#AAAAAAAAGQ) *implementation.*

file [srvAppLayer.h](#AAAAAAAAHZ)

*Class* [*srvAppLayer*](#AAAAAAAAGQ) *interface header.*

# Class Documentation

## ap\_Option Struct Reference

#include <carg\_parser.h>

### Public Attributes

int [code](#AAAAAAAAJH)

const char \* [name](#AAAAAAAAJI)

[ap\_Has\_arg](#AAAAAAAADA) [has\_arg](#AAAAAAAAJJ)

### Detailed Description

Definition at line 44 of file carg\_parser.h.

### Member Data Documentation

#### int [ap\_Option::code](#AAAAAAAAJH)

Definition at line 46 of file carg\_parser.h.

Referenced by optname(), parse\_long\_option(), and parse\_short\_option().

#### const char\* [ap\_Option::name](#AAAAAAAAJI)

Definition at line 47 of file carg\_parser.h.

Referenced by optname().

#### [ap\_Has\_arg](#AAAAAAAADA) [ap\_Option::has\_arg](#AAAAAAAAJJ)

Definition at line 48 of file carg\_parser.h.

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

src/arg\_parser/[carg\_parser.h](#AAAAAAAACY)

## ap\_Record Struct Reference

#include <carg\_parser.h>

### Public Attributes

int [code](#AAAAAAAAJK)

char \* [argument](#AAAAAAAAJL)

### Detailed Description

Definition at line 53 of file carg\_parser.h.

### Member Data Documentation

#### int [ap\_Record::code](#AAAAAAAAJK)

Definition at line 55 of file carg\_parser.h.

Referenced by ap\_code(), and push\_back\_record().

#### char\* [ap\_Record::argument](#AAAAAAAAJL)

Definition at line 56 of file carg\_parser.h.

Referenced by ap\_argument(), free\_data(), and push\_back\_record().

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

src/arg\_parser/[carg\_parser.h](#AAAAAAAACY)

## Arg\_parser Struct Reference

#include <carg\_parser.h>

Collaboration diagram for Arg\_parser:

### Public Attributes

[ap\_Record](#AAAAAAAACZ) \* [data](#AAAAAAAAJM)

char \* [error](#AAAAAAAAJN)

int [data\_size](#AAAAAAAAJO)

int [error\_size](#AAAAAAAAJP)

### Detailed Description

Definition at line 61 of file carg\_parser.h.

### Member Data Documentation

#### [ap\_Record](#AAAAAAAACZ)\* [Arg\_parser::data](#AAAAAAAAJM)

Definition at line 63 of file carg\_parser.h.

Referenced by ap\_argument(), ap\_code(), ap\_init(), free\_data(), and push\_back\_record().

#### char\* [Arg\_parser::error](#AAAAAAAAJN)

Definition at line 64 of file carg\_parser.h.

Referenced by add\_error(), ap\_error(), ap\_free(), and ap\_init().

#### int [Arg\_parser::data\_size](#AAAAAAAAJO)

Definition at line 65 of file carg\_parser.h.

Referenced by ap\_arguments(), ap\_init(), free\_data(), and push\_back\_record().

#### int [Arg\_parser::error\_size](#AAAAAAAAJP)

Definition at line 66 of file carg\_parser.h.

Referenced by add\_error(), ap\_free(), and ap\_init().

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

src/arg\_parser/[carg\_parser.h](#AAAAAAAACY)

## buffer Class Reference

Simple queue of bytes.

#include <buffer.h>

### Public Member Functions

[buffer](#AAAAAAAAJQ) (DWORD size)

*Constructs buffer as bytes array of* ***size*** *.*

[~buffer](#AAAAAAAAJR) ()

errType [write](#AAAAAAAAJS) (BYTE \*addr, DWORD len)

*Try to write to array data from* ***addr*** *pointer with* ***len*** *bytes size.*

DWORD [read](#AAAAAAAAJT) (BYTE \*addr, DWORD len=0)

*Try to read from array data to* ***addr*** *pointer with* ***len*** *bytes size.*

DWORD [length](#AAAAAAAAJU) ()

*Returns length of stored data.*

BYTE \* [lockBufferChunkForExternWriting](#AAAAAAAAJV) ()

*Allow direct mode writing for external methods.*

errType [unlockBufferChunkForExternWriting](#AAAAAAAAJW) (DWORD offset)

*Disable direct mode writing from external methods.*

errType [removeBufferChunk](#AAAAAAAAJX) (DWORD backward\_offset, DWORD len)

*Remove data from buffer.*

errType [copyBufferChunkTo](#AAAAAAAAJY) (BYTE \*addr, DWORD offset=0, DWORD len=0)

*Try to read from array* ***offset*** *data to* ***addr*** *pointer with* ***len*** *bytes size.*

errType [dbgPrint](#AAAAAAAAJZ) ()

*Print for debug purposes contents of stored buffer (in a hexadecimal notation)*

### Private Attributes

BYTE \* [buff](#AAAAAAAAKA)

DWORD [WrRef](#AAAAAAAAKB)

DWORD [RdRef](#AAAAAAAAKC)

DWORD [buffSize](#AAAAAAAAKD)

bool [writingLock](#AAAAAAAAKE)

### Detailed Description

Simple queue of bytes.

Definition at line 20 of file buffer.h.

### Constructor & Destructor Documentation

#### buffer::buffer (DWORD *size*)

Constructs buffer as bytes array of **size** .

Definition at line 23 of file buffer.cpp.

References buff, buffSize, RdRef, writingLock, and WrRef.

#### buffer::~buffer ()

Definition at line 33 of file buffer.cpp.

References buff, RdRef, writingLock, and WrRef.

### Member Function Documentation

#### errType buffer::write (BYTE \* *addr*, DWORD *len*)

Try to write to array data from **addr** pointer with **len** bytes size.

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - writing was succeeded |
| *err\_result\_error* | - not enough space. Data was written partially. |

Definition at line 58 of file buffer.cpp.

References buff, buffSize, and WrRef.

#### DWORD buffer::read (BYTE \* *addr*, DWORD *len* = 0)

Try to read from array data to **addr** pointer with **len** bytes size.

If **len** not specified (or equal to 0) method reads all array.

Readed data has been deleted in buffer.

##### See also:

[buffer::copyBufferChunkTo](#AAAAAAAAJY)

##### Return values:

|  |  |
| --- | --- |
| *length* | - size in bytes of readed data. |

Definition at line 81 of file buffer.cpp.

References buff, length(), RdRef, writingLock, and WrRef.

Here is the call graph for this function:

#### DWORD buffer::length ()

Returns length of stored data.

length is writing reference index minus reading reference index

##### Return values:

|  |  |
| --- | --- |
| *length* | - stored data length in bytes |

Definition at line 48 of file buffer.cpp.

References RdRef, and WrRef.

Referenced by read().

Here is the caller graph for this function:

#### BYTE \* buffer::lockBufferChunkForExternWriting ()

Allow direct mode writing for external methods.

While direct mode writing is locking array - nobody can't to write.

##### See also:

Unlock with [buffer::unlockBufferChunkForExternWriting](#AAAAAAAAJW).

##### Return values:

|  |  |
| --- | --- |
| *ptr* | - pointer to array for writing data |

Definition at line 116 of file buffer.cpp.

References buff, writingLock, and WrRef.

#### errType buffer::unlockBufferChunkForExternWriting (DWORD *offset*)

Disable direct mode writing from external methods.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *offset* | - length of writing bytes by external methods |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - unlocking was successfull |
| *err\_result\_error* | - writing bytes was more than space in buffer. Data unlocked to buffer partially. |

Definition at line 131 of file buffer.cpp.

References buffSize, RdRef, writingLock, and WrRef.

#### errType buffer::removeBufferChunk (DWORD *backward\_offset*, DWORD *len*)

Remove data from buffer.

Removes data from buffer array, backward writing reference to **backward\_offset** and move data from **backward\_offset** + **len** to current writing referense position.

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - removing was successful. |
| *err\_result\_error* | - cannot remove data length more than data size in buffer |

Definition at line 153 of file buffer.cpp.

References buff, RdRef, and WrRef.

#### errType buffer::copyBufferChunkTo (BYTE \* *addr*, DWORD *offset* = 0, DWORD *len* = 0)

Try to read from array **offset** data to **addr** pointer with **len** bytes size.

Extract buffer data without deleting it in buffer.

##### See also:

[buffer::read](#AAAAAAAAJT)

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - successfully copied. |
| *err\_result\_error* | - start of block (**offset** ) is greater than stored data length. |

Definition at line 175 of file buffer.cpp.

References buff, RdRef, and WrRef.

#### errType buffer::dbgPrint ()

Print for debug purposes contents of stored buffer (in a hexadecimal notation)

Definition at line 190 of file buffer.cpp.

References buff, RdRef, and WrRef.

### Member Data Documentation

#### BYTE\* [buffer::buff](#AAAAAAAAKA) [private]

Definition at line 21 of file buffer.h.

Referenced by buffer(), copyBufferChunkTo(), dbgPrint(), lockBufferChunkForExternWriting(), read(), removeBufferChunk(), write(), and ~buffer().

#### DWORD [buffer::WrRef](#AAAAAAAAKB) [private]

Definition at line 22 of file buffer.h.

Referenced by buffer(), copyBufferChunkTo(), dbgPrint(), length(), lockBufferChunkForExternWriting(), read(), removeBufferChunk(), unlockBufferChunkForExternWriting(), write(), and ~buffer().

#### DWORD [buffer::RdRef](#AAAAAAAAKC) [private]

Definition at line 22 of file buffer.h.

Referenced by buffer(), copyBufferChunkTo(), dbgPrint(), length(), read(), removeBufferChunk(), unlockBufferChunkForExternWriting(), and ~buffer().

#### DWORD [buffer::buffSize](#AAAAAAAAKD) [private]

Definition at line 22 of file buffer.h.

Referenced by buffer(), unlockBufferChunkForExternWriting(), and write().

#### bool [buffer::writingLock](#AAAAAAAAKE) [private]

Definition at line 24 of file buffer.h.

Referenced by buffer(), lockBufferChunkForExternWriting(), read(), unlockBufferChunkForExternWriting(), and ~buffer().

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

src/buffer/[buffer.h](#AAAAAAAAEP)

src/buffer/[buffer.cpp](#AAAAAAAAEM)

## commonFuncsMgr Class Reference

common functions manager implementation (set of function independs on target)

#include <commonFuncsMgr.h>

Collaboration diagram for commonFuncsMgr:

### Public Member Functions

[commonFuncsMgr](#AAAAAAAAKF) ([srvAppLayer](#AAAAAAAAGQ) \*appl)

[~commonFuncsMgr](#AAAAAAAAKG) ()

errType [startCommonFuncs](#AAAAAAAAGW) ()

*Declaration of common functions.*

errType [stopCommonFuncs](#AAAAAAAAKH) ()

*Undeclaration of common functions.*

### Private Attributes

[srvAppLayer](#AAAAAAAAGQ) \* [appLayer](#AAAAAAAAKI)

### Detailed Description

common functions manager implementation (set of function independs on target)

Definition at line 23 of file commonFuncsMgr.h.

### Constructor & Destructor Documentation

#### commonFuncsMgr::commonFuncsMgr ([srvAppLayer](#AAAAAAAAGQ) \* *appl*)

Definition at line 60 of file commonFuncsMgr.cpp.

References appLayer.

#### commonFuncsMgr::~commonFuncsMgr ()

Definition at line 65 of file commonFuncsMgr.cpp.

### Member Function Documentation

#### errType commonFuncsMgr::startCommonFuncs ()

Declaration of common functions.

Declares functions:

emergencyShutdown , controlModeChange , getStateVector

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - declaration was successful |

Definition at line 76 of file commonFuncsMgr.cpp.

References appLayer, srvAppLayer::CreateNewFunction(), functionNode::setFuncName(), functionNode::setParamDescriptor(), functionNode::setParamName(), functionNode::setResultDescriptor(), and functionNode::setResultName().

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType commonFuncsMgr::stopCommonFuncs ()

Undeclaration of common functions.

Undeclares functions:

emergencyShutdown , controlModeChange , getStateVector

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - undeclaration was successful |

Definition at line 151 of file commonFuncsMgr.cpp.

References appLayer, and srvAppLayer::DeleteFunction().

Here is the call graph for this function:

### Member Data Documentation

#### [srvAppLayer](#AAAAAAAAGQ)\* [commonFuncsMgr::appLayer](#AAAAAAAAKI) [private]

Definition at line 25 of file commonFuncsMgr.h.

Referenced by commonFuncsMgr(), startCommonFuncs(), and stopCommonFuncs().

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

src/functions/[commonFuncsMgr.h](#AAAAAAAAGF)

src/functions/[commonFuncsMgr.cpp](#AAAAAAAAGB)

## deqUdp Class Reference

udp communications (based on udp\_port) with queues for listening and sending

#include <deqUdp.h>

### Public Member Functions

[deqUdp](#AAAAAAAAKJ) (WORD portNum, const char \*ip\_str="127.0.0.1")

[~deqUdp](#AAAAAAAAKK) ()

errType [sendData](#AAAAAAAAIL) ([ssBuffer](#AAAAAAAAES) \*buf)

*Send data to udp port from queued buffer (*[*ssBuffer*](#AAAAAAAAES)*)*

errType [readData](#AAAAAAAAIJ) ([ssBuffer](#AAAAAAAAES) \*addr, size\_t \*len, in\_addr \*ip\_from=0, bool peek=false)

*Read udp port data to queued buffer (*[*ssBuffer*](#AAAAAAAAES)*)*

### Detailed Description

udp communications (based on udp\_port) with queues for listening and sending

Definition at line 20 of file deqUdp.h.

### Constructor & Destructor Documentation

#### deqUdp::deqUdp (WORD *portNum*, const char \* *ip\_str* = "127.0.0.1") [inline]

Definition at line 27 of file deqUdp.h.

#### deqUdp::~deqUdp ()

Definition at line 28 of file deqUdp.cpp.

### Member Function Documentation

#### errType deqUdp::sendData ([ssBuffer](#AAAAAAAAES) \* *buf*)

Send data to udp port from queued buffer ([ssBuffer](#AAAAAAAAES))

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *buf* | - pointer to queue for sending data |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - execution was successful |
| *err\_result\_error* | - problems with udp sendto function |

##### [Todo](#dd/da0/todo__todo000008):

not needed to realize all sendData check ability to use base method sendData of parent class

Definition at line 39 of file deqUdp.cpp.

References ssBuffer::getFrontBlockSize(), and ssBuffer::popBlock().

Referenced by udpSenderThread().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType deqUdp::readData ([ssBuffer](#AAAAAAAAES) \* *addr*, size\_t \* *len*, in\_addr \* *ipaddr* = 0, bool *peek* = false)

Read udp port data to queued buffer ([ssBuffer](#AAAAAAAAES))

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *addr* | - pointer to queue for readed data |
| in | *peek* | - PEEK MODE |
| out | *len* | - length of readed data |
| out | *ipaddr* | - information about data sender |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - execution was successful |
| *err\_result\_error* | - problems with udp sendto function |

##### [Todo](#dd/da0/todo__todo000009):

not needed to realize all sendData check ability to use base method sendData of parent class

Definition at line 77 of file deqUdp.cpp.

References ssBuffer::pushBlock().

Referenced by udpListenerThread().

Here is the call graph for this function:

Here is the caller graph for this function:

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

src/deqUdp/[deqUdp.h](#AAAAAAAAFZ)

src/deqUdp/[deqUdp.cpp](#AAAAAAAAFW)

## functionNode Class Reference

function node interface header

#include <functionNode.h>

Collaboration diagram for functionNode:

### Public Member Functions

[functionNode](#AAAAAAAAKL) (BYTE id, WORD parQnt, WORD resQnt, funcPtr ptr)

[*functionNode*](#AAAAAAAAHI) *constructor.*

[~functionNode](#AAAAAAAAKM) ()

errType [setFuncName](#AAAAAAAAKN) (const char \*name)

*Setter for function node string name.*

errType [setParamDescriptor](#AAAAAAAAKO) (BYTE num, OrtsType type)

*Method declares parameter for function node.*

errType [setResultDescriptor](#AAAAAAAAKP) (BYTE num, OrtsType type)

*Method declares results for function node.*

errType [setParamName](#AAAAAAAAKQ) (BYTE num, const char \*name)

*Setter for function node parameter string name.*

errType [setResultName](#AAAAAAAAKR) (BYTE num, const char \*name)

*Setter for function node result string name.*

WORD [getParamLength](#AAAAAAAAKS) (BYTE num)

*Method evaluate function node parameter size.*

OrtsType [getParamType](#AAAAAAAAKT) (BYTE num)

WORD [getParamsQuantity](#AAAAAAAAKU) ()

*Getter for function parameters quantity.*

WORD [getAllParamsLength](#AAAAAAAAKV) ()

*Method calculate sum of all function node parameters size.*

errType [decodeParams](#AAAAAAAAKW) (rcsCmd \*)

*Received message parameters decode.*

errType [setParam](#AAAAAAAAKX) (BYTE num, void \*param)

*Set value for declared parameter.*

void \* [getParamPtr](#AAAAAAAAKY) (BYTE num)

*Get value pointer of declared parameter.*

BYTE [getResultsQuantity](#AAAAAAAAKZ) ()

*Getter for function results quantity.*

DWORD [getAllResultsLength](#AAAAAAAALA) ()

*Method calculate sum of all function node results size.*

DWORD [getResultLength](#AAAAAAAALB) (BYTE i)

*Method evaluate function node result size.*

OrtsType [getResultType](#AAAAAAAALC) (BYTE num)

errType [getResult](#AAAAAAAALD) (BYTE num, void \*\*result, DWORD \*length)

*Getter function for function node result storage.*

errType [setResult](#AAAAAAAALE) (BYTE num, void \*result)

*Setter function for function node result storage.*

errType [printParams](#AAAAAAAALF) ()

*Prints params values of function node for debug purposes.*

errType [printResults](#AAAAAAAALG) ()

*Prints results values of function node for debug purposes.*

errType [callFunction](#AAAAAAAALH) ()

*Call function node pointer to execution code.*

BYTE [id](#AAAAAAAALI) ()

*Getter for function node identifier.*

void [dbgPrint](#AAAAAAAALJ) ()

*Print for debug purposes contents of function node (in a hexadecimal notation)*

### Private Attributes

BYTE [func\_id](#AAAAAAAALK)

*function idintifier.*

const char \* [func\_name](#AAAAAAAALL)

*function name. For debug printing purposes.*

WORD [func\_paramsQuantity](#AAAAAAAALM)

*function declaration. Quantity of params.*

WORD [func\_resultsQuantity](#AAAAAAAALN)

*function declaration. Quantity of results.*

WORD [func\_paramsLength](#AAAAAAAALO)

*Size in bytes of all function parameters.*

WORD [func\_resultsLength](#AAAAAAAALP)

*Size in bytes of all function reaults.*

[param\_desc](#AAAAAAAAHN) \* [func\_params](#AAAAAAAALQ) [32]

*function parameters declarations. 32 parameters at maximum.*

[param\_desc](#AAAAAAAAHN) \* [func\_results](#AAAAAAAALR) [32]

*function results declaration. 32 results at maximum.*

funcPtr [func](#AAAAAAAALS)

*pointer to a function execution code.*

### Detailed Description

function node interface header

function node is a service function that will call by [srvAppLayer](#AAAAAAAAGQ) instance in accordance with client request

Definition at line 20 of file functionNode.h.

### Constructor & Destructor Documentation

#### functionNode::functionNode (BYTE *id*, WORD *parQnt*, WORD *resQnt*, funcPtr *ptr*)

[functionNode](#AAAAAAAAHI) constructor.

uses to declare service function.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *id* | - function identifier. |
| in | *parQnt* | - quantity of function parameters. |
| in | *resQnt* | - quantity of function results. |
| in | *ptr* | - function execution code pointer. |

Definition at line 29 of file functionNode.cpp.

References func, func\_id, func\_params, func\_paramsLength, func\_paramsQuantity, func\_resultsLength, func\_resultsQuantity, and id().

Here is the call graph for this function:

#### functionNode::~functionNode ()

Definition at line 42 of file functionNode.cpp.

References func\_params, func\_paramsQuantity, func\_results, and func\_resultsQuantity.

### Member Function Documentation

#### errType functionNode::setFuncName (const char \* *name*)

Setter for function node string name.

##### Parameters:

|  |  |
| --- | --- |
| *name* | - name of a function node. |

##### Returns:

err\_result\_ok

Definition at line 362 of file functionNode.cpp.

References func\_name.

Referenced by commonFuncsMgr::startCommonFuncs().

Here is the caller graph for this function:

#### errType functionNode::setParamDescriptor (BYTE *num*, OrtsType *type*)

Method declares parameter for function node.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - parameter index (zero based index) |
| *type* | - parameter type with respect to OrtsType enumeration |

##### Returns:

err\_result\_ok - declaring successful

Definition at line 61 of file functionNode.cpp.

References func\_params, func\_paramsLength, func\_paramsQuantity, and param\_desc::length().

Referenced by commonFuncsMgr::startCommonFuncs().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType functionNode::setResultDescriptor (BYTE *num*, OrtsType *type*)

Method declares results for function node.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - result index (zero based index) |
| *type* | - result type with respect to OrtsType enumeration |

##### Returns:

err\_result\_ok - declaring successful err\_not\_init - declaring unsuccessful

Definition at line 83 of file functionNode.cpp.

References func\_results.

Referenced by commonFuncsMgr::startCommonFuncs().

Here is the caller graph for this function:

#### errType functionNode::setParamName (BYTE *num*, const char \* *name*)

Setter for function node parameter string name.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - number of a parameter |
| *name* | - name of a parameter |

##### Returns:

err\_result\_ok

Definition at line 377 of file functionNode.cpp.

References func\_params, and param\_desc::setName().

Referenced by commonFuncsMgr::startCommonFuncs().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType functionNode::setResultName (BYTE *num*, const char \* *name*)

Setter for function node result string name.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - number of a result |
| *name* | - name of a result |

##### Returns:

err\_result\_ok

Definition at line 392 of file functionNode.cpp.

References func\_results, and param\_desc::setName().

Referenced by commonFuncsMgr::startCommonFuncs().

Here is the call graph for this function:

Here is the caller graph for this function:

#### WORD functionNode::getParamLength (BYTE *num*)

Method evaluate function node parameter size.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - parameter index (zero based index) |

##### Returns:

size of parameter in bytes

Definition at line 105 of file functionNode.cpp.

References func\_params, and func\_paramsQuantity.

#### OrtsType functionNode::getParamType (BYTE *num*)

#### WORD functionNode::getParamsQuantity ()

Getter for function parameters quantity.

Definition at line 51 of file functionNode.cpp.

References func\_paramsQuantity.

#### WORD functionNode::getAllParamsLength ()

Method calculate sum of all function node parameters size.

##### Returns:

size of all parameters in bytes

Definition at line 117 of file functionNode.cpp.

References func\_params, func\_paramsLength, func\_paramsQuantity, and param\_desc::length().

Here is the call graph for this function:

#### errType functionNode::decodeParams (rcsCmd \* *packet*)

Received message parameters decode.

check parameters quantity and fills parameters pointers with received values

##### Parameters:

|  |  |
| --- | --- |
| *packet* | - received rcsCmd message |

##### Returns:

err\_result\_error - received parameters does not match to declaration

err\_result\_ok - received parameters successfully decodes

1. Define length of parametric part by function declaration

2. Count length of parametric part of received request message

3. Compare received params quantity with declared params quantity

4. If received params quantity is equal to declared params quantity - fills parameters values. otherwise - return value is err\_result\_error.

Definition at line 191 of file functionNode.cpp.

References func\_params, func\_paramsLength, func\_paramsQuantity, param\_desc::length(), and param\_desc::setParam().

Here is the call graph for this function:

#### errType functionNode::setParam (BYTE *num*, void \* *param*)

Set value for declared parameter.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - parameter index (zero based index) |
| *param* | - pointer to value |

##### Returns:

err\_result\_ok - value setting was successful

Definition at line 165 of file functionNode.cpp.

References func\_params, and param\_desc::setParam().

Here is the call graph for this function:

#### void \* functionNode::getParamPtr (BYTE *num*)

Get value pointer of declared parameter.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - parameter index (zero based index) |

##### Returns:

nonzero - value pointer

0 - parameter not exist

Definition at line 178 of file functionNode.cpp.

References func\_params, and param\_desc::value().

Here is the call graph for this function:

#### BYTE functionNode::getResultsQuantity ()

Getter for function results quantity.

Definition at line 153 of file functionNode.cpp.

References func\_resultsQuantity.

#### DWORD functionNode::getAllResultsLength ()

Method calculate sum of all function node results size.

##### Returns:

size of all results in bytes

Definition at line 131 of file functionNode.cpp.

References func\_results, func\_resultsLength, func\_resultsQuantity, and param\_desc::length().

Here is the call graph for this function:

#### DWORD functionNode::getResultLength (BYTE *res\_no*)

Method evaluate function node result size.

##### Parameters:

|  |  |
| --- | --- |
| *num* | - result index (zero based index) |

##### Returns:

size of result in bytes

Definition at line 145 of file functionNode.cpp.

References func\_results, and param\_desc::length().

Here is the call graph for this function:

#### OrtsType functionNode::getResultType (BYTE *num*)

#### errType functionNode::getResult (BYTE *num*, void \*\* *out\_res*, DWORD \* *length*)

Getter function for function node result storage.

##### Parameters:

|  |  |
| --- | --- |
| *num[in]* | - number of result in declaration of function node |
| *out\_res[out]* | - pointer to result value |
| *length[out]* | - length in bytes of stored value |

##### Returns:

err\_result\_ok - value gets successfully

Definition at line 280 of file functionNode.cpp.

References func\_results, param\_desc::length(), and param\_desc::value().

Here is the call graph for this function:

#### errType functionNode::setResult (BYTE *num*, void \* *res*)

Setter function for function node result storage.

##### Parameters:

|  |  |
| --- | --- |
| *num[in]* | - number of result in declaration of function node |
| *res[in]* | - pointer to result value |

##### Returns:

err\_result\_ok - value successfully sets

Definition at line 261 of file functionNode.cpp.

References func\_id, func\_results, and param\_desc::setParam().

Here is the call graph for this function:

#### errType functionNode::printParams ()

Prints params values of function node for debug purposes.

##### Returns:

err\_result\_ok - values prints successfully

Definition at line 293 of file functionNode.cpp.

References func\_params, func\_paramsQuantity, and param\_desc::printParam().

Here is the call graph for this function:

#### errType functionNode::printResults ()

Prints results values of function node for debug purposes.

##### Returns:

err\_result\_ok - values prints successfully

Definition at line 316 of file functionNode.cpp.

References func\_results, func\_resultsQuantity, and param\_desc::printParam().

Here is the call graph for this function:

#### errType functionNode::callFunction ()

Call function node pointer to execution code.

##### Returns:

result - result code returns from calling code as return value

Definition at line 338 of file functionNode.cpp.

References func, and func\_name.

#### BYTE functionNode::id ()

Getter for function node identifier.

##### Returns:

func\_id - function identifier

Definition at line 352 of file functionNode.cpp.

References func\_id.

Referenced by srvAppLayer::CreateNewFunction(), and functionNode().

Here is the caller graph for this function:

#### void functionNode::dbgPrint ()

Print for debug purposes contents of function node (in a hexadecimal notation)

Definition at line 404 of file functionNode.cpp.

References func\_id, func\_params, func\_paramsLength, func\_paramsQuantity, and func\_resultsQuantity.

### Member Data Documentation

#### BYTE [functionNode::func\_id](#AAAAAAAALK) [private]

function idintifier.

Definition at line 22 of file functionNode.h.

Referenced by dbgPrint(), functionNode(), id(), and setResult().

#### const char\* [functionNode::func\_name](#AAAAAAAALL) [private]

function name. For debug printing purposes.

Definition at line 23 of file functionNode.h.

Referenced by callFunction(), and setFuncName().

#### WORD [functionNode::func\_paramsQuantity](#AAAAAAAALM) [private]

function declaration. Quantity of params.

Definition at line 24 of file functionNode.h.

Referenced by dbgPrint(), decodeParams(), functionNode(), getAllParamsLength(), getParamLength(), getParamsQuantity(), printParams(), setParamDescriptor(), and ~functionNode().

#### WORD [functionNode::func\_resultsQuantity](#AAAAAAAALN) [private]

function declaration. Quantity of results.

Definition at line 25 of file functionNode.h.

Referenced by dbgPrint(), functionNode(), getAllResultsLength(), getResultsQuantity(), printResults(), and ~functionNode().

#### WORD [functionNode::func\_paramsLength](#AAAAAAAALO) [private]

Size in bytes of all function parameters.

Definition at line 27 of file functionNode.h.

Referenced by dbgPrint(), decodeParams(), functionNode(), getAllParamsLength(), and setParamDescriptor().

#### WORD [functionNode::func\_resultsLength](#AAAAAAAALP) [private]

Size in bytes of all function reaults.

Definition at line 28 of file functionNode.h.

Referenced by functionNode(), and getAllResultsLength().

#### [param\_desc](#AAAAAAAAHN)\* [functionNode::func\_params](#AAAAAAAALQ)[32] [private]

function parameters declarations. 32 parameters at maximum.

Definition at line 30 of file functionNode.h.

Referenced by dbgPrint(), decodeParams(), functionNode(), getAllParamsLength(), getParamLength(), getParamPtr(), printParams(), setParam(), setParamDescriptor(), setParamName(), and ~functionNode().

#### [param\_desc](#AAAAAAAAHN)\* [functionNode::func\_results](#AAAAAAAALR)[32] [private]

function results declaration. 32 results at maximum.

Definition at line 31 of file functionNode.h.

Referenced by getAllResultsLength(), getResult(), getResultLength(), printResults(), setResult(), setResultDescriptor(), setResultName(), and ~functionNode().

#### funcPtr [functionNode::func](#AAAAAAAALS) [private]

pointer to a function execution code.

Definition at line 33 of file functionNode.h.

Referenced by callFunction(), and functionNode().

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

src/srvAppLayer/functionNode/[functionNode.h](#AAAAAAAAHK)

src/srvAppLayer/functionNode/[functionNode.cpp](#AAAAAAAAHH)

## param\_desc Class Reference

parameter description

#include <param\_desc.h>

### Public Member Functions

[param\_desc](#AAAAAAAALT) (OrtsType type, WORD len)

*Constructs parameter description with some type and information about storing value length.*

[param\_desc](#AAAAAAAALU) (OrtsType type)

*Overloaded constructor that constructs parameter description of a vector type.*

[~param\_desc](#AAAAAAAALV) ()

errType [resize](#AAAAAAAALW) (WORD new\_size)

*Function calling if need to resize of storage for new value or something else...*

void \* [value](#AAAAAAAALX) ()

*Getter for pointer of storing value.*

WORD [length](#AAAAAAAALY) ()

*Getter for length of storing value.*

OrtsType [type](#AAAAAAAALZ) ()

*Getter for type of storing value.*

bool [isVector](#AAAAAAAAMA) ()

*Function checks if type of storing value is a vector.*

errType [setParam](#AAAAAAAAMB) (void \*param\_val)

*Setter function for setting param value.*

errType [setName](#AAAAAAAAMC) (const char \*name)

*Setter for parameter string name.*

errType [printParam](#AAAAAAAAMD) ()

*Print parameter value for debug purposes.*

void [dbgPrint](#AAAAAAAAME) ()

*Print for debug purposes contents of function node (in a hexadecimal notation)*

### Private Attributes

void \* [param](#AAAAAAAAMF)

*pointer to parameter value*

char \* [param\_name](#AAAAAAAAMG)

*parameter string name*

WORD [\_length](#AAAAAAAAMH)

*length in bytes of storing parameter value*

OrtsType [\_type](#AAAAAAAAMI)

bool [\_isVector](#AAAAAAAAMJ)

*storage value is a vector (one-dimensional array) of simplified values types*

### Detailed Description

parameter description

parameter description uses in function node parameter declaration

Definition at line 20 of file param\_desc.h.

### Constructor & Destructor Documentation

#### param\_desc::param\_desc (OrtsType *type*, WORD *param\_len*)

Constructs parameter description with some type and information about storing value length.

##### Parameters:

|  |  |
| --- | --- |
| *type* | - type (or simplified reference for vector) of storing value |
| *param\_len* | - length in bytes of storing value |

##### Returns:

Definition at line 26 of file param\_desc.cpp.

References \_isVector, \_length, \_type, param, and type().

Here is the call graph for this function:

#### param\_desc::param\_desc (OrtsType *type*)

Overloaded constructor that constructs parameter description of a vector type.

##### Parameters:

|  |  |
| --- | --- |
| *type* | - type of storing value |

##### Returns:

Definition at line 40 of file param\_desc.cpp.

References \_isVector, \_length, \_type, and type().

Here is the call graph for this function:

#### param\_desc::~param\_desc ()

Definition at line 49 of file param\_desc.cpp.

References param, and param\_name.

### Member Function Documentation

#### errType param\_desc::resize (WORD *new\_size*)

Function calling if need to resize of storage for new value or something else...

##### Parameters:

|  |  |
| --- | --- |
| *new\_size* | - size in bytes of new storage |

##### Returns:

err\_result\_ok - memory successfully reallocated

err\_mem\_alloc - error of allocating memory for new size

err\_not\_init - new size does not differ from old size

Definition at line 63 of file param\_desc.cpp.

References \_length, and param.

Referenced by setParam().

Here is the caller graph for this function:

#### void \* param\_desc::value ()

Getter for pointer of storing value.

##### Returns:

param - pointer to storing value

Definition at line 92 of file param\_desc.cpp.

References param.

Referenced by functionNode::getParamPtr(), and functionNode::getResult().

Here is the caller graph for this function:

#### WORD param\_desc::length ()

Getter for length of storing value.

##### Returns:

length - storage size in bytes

Definition at line 83 of file param\_desc.cpp.

References \_length.

Referenced by functionNode::decodeParams(), functionNode::getAllParamsLength(), functionNode::getAllResultsLength(), functionNode::getResult(), functionNode::getResultLength(), and functionNode::setParamDescriptor().

Here is the caller graph for this function:

#### OrtsType param\_desc::type ()

Getter for type of storing value.

##### Returns:

type - pointer to storing value type

Definition at line 101 of file param\_desc.cpp.

References \_type.

Referenced by param\_desc().

Here is the caller graph for this function:

#### bool param\_desc::isVector ()

Function checks if type of storing value is a vector.

##### Returns:

true - storing value is a vector

false - storing value is a scalar

Definition at line 111 of file param\_desc.cpp.

References \_type.

#### errType param\_desc::setParam (void \* *param\_val*)

Setter function for setting param value.

##### Parameters:

|  |  |
| --- | --- |
| *param\_val[in]* | - pointer to parameter value |

##### Returns:

err\_result\_ok - value was successfully set

err\_params\_value - null-pointer error

Definition at line 122 of file param\_desc.cpp.

References \_isVector, \_length, param, and resize().

Referenced by functionNode::decodeParams(), functionNode::setParam(), and functionNode::setResult().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType param\_desc::setName (const char \* *name*)

Setter for parameter string name.

##### Parameters:

|  |  |
| --- | --- |
| *name* | - name of a parameter. |

##### Returns:

err\_result\_ok

Definition at line 139 of file param\_desc.cpp.

References param\_name.

Referenced by functionNode::setParamName(), and functionNode::setResultName().

Here is the caller graph for this function:

#### errType param\_desc::printParam ()

Print parameter value for debug purposes.

##### Returns:

err\_result\_ok - value was printed successfully

Definition at line 151 of file param\_desc.cpp.

References \_type, param, and param\_name.

Referenced by functionNode::printParams(), and functionNode::printResults().

Here is the caller graph for this function:

#### void param\_desc::dbgPrint ()

Print for debug purposes contents of function node (in a hexadecimal notation)

Definition at line 207 of file param\_desc.cpp.

References param.

### Member Data Documentation

#### void\* [param\_desc::param](#AAAAAAAAMF) [private]

pointer to parameter value

Definition at line 22 of file param\_desc.h.

Referenced by dbgPrint(), param\_desc(), printParam(), resize(), setParam(), value(), and ~param\_desc().

#### char\* [param\_desc::param\_name](#AAAAAAAAMG) [private]

parameter string name

Definition at line 23 of file param\_desc.h.

Referenced by printParam(), setName(), and ~param\_desc().

#### WORD [param\_desc::\_length](#AAAAAAAAMH) [private]

length in bytes of storing parameter value

Definition at line 24 of file param\_desc.h.

Referenced by length(), param\_desc(), resize(), and setParam().

#### OrtsType [param\_desc::\_type](#AAAAAAAAMI) [private]

Definition at line 25 of file param\_desc.h.

Referenced by isVector(), param\_desc(), printParam(), and type().

#### bool [param\_desc::\_isVector](#AAAAAAAAMJ) [private]

storage value is a vector (one-dimensional array) of simplified values types

Definition at line 27 of file param\_desc.h.

Referenced by param\_desc(), and setParam().

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

src/srvAppLayer/functionNode/[param\_desc.h](#AAAAAAAAHP)

src/srvAppLayer/functionNode/[param\_desc.cpp](#AAAAAAAAHM)

## serviceState Struct Reference

[stateVector\_type](#AAAAAAAAIB) structural field.

#include <srvAppLayer.h>

### Public Attributes

BYTE [linked](#AAAAAAAAMK):1

BYTE [reserved](#AAAAAAAAML):3

BYTE [inprocess](#AAAAAAAAMM):1

BYTE [mode\_emergency](#AAAAAAAAMN):1

BYTE [mode\_manual](#AAAAAAAAMO):1

BYTE [mode\_auto](#AAAAAAAAMP):1

### Detailed Description

[stateVector\_type](#AAAAAAAAIB) structural field.

Definition at line 21 of file srvAppLayer.h.

### Member Data Documentation

#### BYTE [serviceState::linked](#AAAAAAAAMK)

Definition at line 22 of file srvAppLayer.h.

#### BYTE [serviceState::reserved](#AAAAAAAAML)

Definition at line 23 of file srvAppLayer.h.

#### BYTE [serviceState::inprocess](#AAAAAAAAMM)

Definition at line 24 of file srvAppLayer.h.

#### BYTE [serviceState::mode\_emergency](#AAAAAAAAMN)

Definition at line 25 of file srvAppLayer.h.

#### BYTE [serviceState::mode\_manual](#AAAAAAAAMO)

Definition at line 26 of file srvAppLayer.h.

#### BYTE [serviceState::mode\_auto](#AAAAAAAAMP)

Definition at line 27 of file srvAppLayer.h.

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

src/srvAppLayer/[srvAppLayer.h](#AAAAAAAAHZ)

## specFuncsMgr Class Reference

special functions set manager.

#include <specFuncsMgr.h>

Collaboration diagram for specFuncsMgr:

### Public Member Functions

[specFuncsMgr](#AAAAAAAAMQ) ([srvAppLayer](#AAAAAAAAGQ) \*appl)

*Links special functions set with* [*srvAppLayer*](#AAAAAAAAGQ)*.*

[~specFuncsMgr](#AAAAAAAAMR) ()

errType [startSpecFuncs](#AAAAAAAAGX) ()

*Declare special functions set.*

errType [stopSpecFuncs](#AAAAAAAAMS) ()

### Private Attributes

[srvAppLayer](#AAAAAAAAGQ) \* [appLayer](#AAAAAAAAMT)

### Detailed Description

special functions set manager.

implementation in /funcs/0/0\_SpecFuncs.cpp ... /funcs/9/9\_CommonFuncs.cpp (set of function depends on target)

Definition at line 20 of file specFuncsMgr.h.

### Constructor & Destructor Documentation

#### specFuncsMgr::specFuncsMgr ([srvAppLayer](#AAAAAAAAGQ) \* *appl*)

Links special functions set with [srvAppLayer](#AAAAAAAAGQ).

#### specFuncsMgr::~specFuncsMgr ()

### Member Function Documentation

#### errType specFuncsMgr::startSpecFuncs ()

Declare special functions set.

Referenced by main().

Here is the caller graph for this function:

#### errType specFuncsMgr::stopSpecFuncs ()

### Member Data Documentation

#### [srvAppLayer](#AAAAAAAAGQ)\* [specFuncsMgr::appLayer](#AAAAAAAAMT) [private]

Definition at line 22 of file specFuncsMgr.h.

#### The documentation for this class was generated from the following file:

src/functions/[specFuncsMgr.h](#AAAAAAAAGI)

## srvAppLayer Class Reference

Application core layer implementaion.

#include <srvAppLayer.h>

Collaboration diagram for srvAppLayer:

### Public Member Functions

[srvAppLayer](#AAAAAAAAMU) (WORD portNum)

*Constructs instance of* [*srvAppLayer*](#AAAAAAAAGQ) *and links it to udp port number for listening clients requests.*

[~srvAppLayer](#AAAAAAAAMV) ()

errType [CreateNewFunction](#AAAAAAAAMW) ([functionNode](#AAAAAAAAHI) \*func)

*Method to add new declared function to service layer.*

errType [DeleteFunction](#AAAAAAAAMX) (BYTE id)

*Method to delete function from service layer.*

errType [encodeBlock](#AAAAAAAAIW) (rcsCmd \*, BYTE \*\*)

*Method to encode data from rcsCmd message.*

errType [StartListening](#AAAAAAAAGU) ()

*Method to prepare and start base communication engine.*

errType [StopListening](#AAAAAAAAMY) ()

*Method to stop base communication engine.*

errType [equip\_reading\_event](#AAAAAAAAJC) ()

*Method to asynchonous polling of* [*equip\_listen*](#AAAAAAAAMZ) *socket.*

errType [equip\_read\_data](#AAAAAAAAJA) (BYTE \*, size\_t \*)

*Method to read data from* [*equip\_listen*](#AAAAAAAAMZ) *socket.*

errType [ProcessMessages](#AAAAAAAAGY) ()

*Method to make one step of* [*srvAppLayer*](#AAAAAAAAGQ) *step.*

WORD [getListenerPortNum](#AAAAAAAANA) ()

*getter for udp port number that listens all clients requests*

BYTE [terminated](#AAAAAAAAGZ) ()

*Method to check application internal termination signal.*

void [terminate](#AAAAAAAANB) (BYTE mode=1)

*getter for AppTerminated signal*

[stateVector\_type](#AAAAAAAAIB) [getStateVector](#AAAAAAAANC) ()

*getter for ServiceState vector*

### Public Attributes

[ssBuffer](#AAAAAAAAES) \* [clientsRequestsQueue](#AAAAAAAAND)

*Queue that stores received requests from client.*

[ssBuffer](#AAAAAAAAES) \* [functionsAnswersQueue](#AAAAAAAANE)

*Queue that stores service functions answers to clients.*

### Private Member Functions

errType [decodeMessage](#AAAAAAAANF) (BYTE \*dataBlock, DWORD length, rcsCmd \*ss\_cmd)

*step 1. decode recieved message from client*

errType [execMessage](#AAAAAAAAJE) (rcsCmd \*ss\_cmd)

*step 2. send data to requested service function*

errType [encodeFuncResult](#AAAAAAAAIY) (rcsCmd \*in\_cmd, rcsCmd \*out\_cmd)

*step 3. encode function execution results for sending back to client*

errType [sendResult](#AAAAAAAANG) (sockaddr\_in \*sin, rcsCmd \*ss\_cmd)

*step 4. send function answer to client*

### Private Attributes

BYTE [AppTerminated](#AAAAAAAANH)

*Application termination process indicator*

0 - Application run normally

1 - Applcation need to exit only

2 - Application need exit with reboot.

[functionNode](#AAAAAAAAHI) \* [Functions](#AAAAAAAAJG) [100]

*Service functions array.*

BYTE [func\_quantity](#AAAAAAAANI)

*Counter that stores really declared functions quantity.*

BYTE [ifCount](#AAAAAAAANJ)

*Counter of ethernet interfaces. No have an idea how to use it.*

pthread\_t [listenerThread](#AAAAAAAANK)

*Handle to client requests listening thread.*

pthread\_t [senderThread](#AAAAAAAANL)

*Handle to client answers sending thread.*

pthread\_t [equipListenThread](#AAAAAAAANM)

*Handle to equipmnent data listening thread.*

WORD [cpListenerPortNum](#AAAAAAAANN)

*settings: Udp port number to listen requests from network clients*

udp\_port \* [equip\_listen](#AAAAAAAAMZ)

*udp\_port instance that associates with listening data from equipment*

[stateVector\_type](#AAAAAAAAIB) [ServiceState](#AAAAAAAANO)

*Service state vector.*

### Detailed Description

Application core layer implementaion.

This layer delegate network calls to service functions and return back functions results.

Definition at line 46 of file srvAppLayer.h.

### Constructor & Destructor Documentation

#### srvAppLayer::srvAppLayer (WORD *portNum*)

Constructs instance of [srvAppLayer](#AAAAAAAAGQ) and links it to udp port number for listening clients requests.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *portNum* | - udp port number that will use for clients requests listening |

Definition at line 159 of file srvAppLayer.cpp.

References AppTerminated, cpListenerPortNum, func\_quantity, and ifCount.

#### srvAppLayer::~srvAppLayer ()

Definition at line 168 of file srvAppLayer.cpp.

References equip\_listen, func\_quantity, and Functions.

### Member Function Documentation

#### errType srvAppLayer::decodeMessage (BYTE \* *dataBlock*, DWORD *length*, rcsCmd \* *ss\_cmd*) [private]

step 1. decode recieved message from client

Method to decode message data to function call and parameters set.

Method using method rcsCmd::encode to transform data array to rcsCmd message.

Method checks for correct sign, calling function existing and compares real received data length with header information about data length.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *dataBlock* | - pointer to received bytes array |
| in | *length* | - size of received bytes array. |
| out | *ss\_cmd* | - decoded message. |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - decoded message is correct. |
| *err\_params\_decode* | - params in decoded message is incorrect. |
| *err\_not\_found* | - decoded message is calling for non existing function |
| *err\_crc\_error* | - decoded message signature is incorrect. |

Definition at line 333 of file srvAppLayer.cpp.

References Functions.

Referenced by ProcessMessages().

Here is the caller graph for this function:

#### errType srvAppLayer::execMessage (rcsCmd \* *ss\_cmd*) [private]

step 2. send data to requested service function

Method to execute function by id in rcsCmd message.

Method checks for correct params by comparing params with description in [commonFuncsMgr](#AAAAAAAAGC) or [specFuncsMgr](#AAAAAAAAGJ).

##### After requested function calling method changes ServiceState vector and set answer ticket for return value from requested function. [Todo](#dd/da0/todo__todo000018):

ServiceState vector need to change before function calling.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *ss\_cmd* | - message with request for function calling. |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - message executed successfully. |
| *err\_params\_decode* | - params in message differs with description. |

Definition at line 383 of file srvAppLayer.cpp.

References Functions, stateVector\_type::lastFuncId, stateVector\_type::lastResult, and ServiceState.

Referenced by ProcessMessages().

Here is the caller graph for this function:

#### errType srvAppLayer::encodeFuncResult (rcsCmd \* *in\_cmd*, rcsCmd \* *out\_cmd*) [private]

step 3. encode function execution results for sending back to client

Method encoding all executed function results to rcsCmd message.

##### Method preparing rcsCmd message with making of message signing. [Todo](#dd/da0/todo__todo000019):

Need a refactoring.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *in\_cmd* | - message with request for function calling. |
| out | *out\_cmd* | - message with results from requested function. |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - message executed successfully. |
| *err\_not\_found* | - function to execute was not found. |

Definition at line 413 of file srvAppLayer.cpp.

References Functions.

Referenced by ProcessMessages().

Here is the caller graph for this function:

#### errType srvAppLayer::sendResult (sockaddr\_in \* *sfrom*, rcsCmd \* *ss\_cmd*) [private]

step 4. send function answer to client

Method sending rcsCmd message to needed recepient.

Method preparing data array from rcsCmd message and push data array with address of recepient in to functionsAnswersQueue

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *sfrom* | - recepient address. |
| in | *ss\_cmd* | - message needed to send. |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - message added to sending queue successfully. |

Definition at line 462 of file srvAppLayer.cpp.

References functionsAnswersQueue, ssBuffer::pushBlock(), and wUdp.

Referenced by ProcessMessages().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType srvAppLayer::CreateNewFunction ([functionNode](#AAAAAAAAHI) \* *func*)

Method to add new declared function to service layer.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *func* | - [functionNode](#AAAAAAAAHI) instance |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - function added successfully |

Definition at line 182 of file srvAppLayer.cpp.

References func\_quantity, Functions, and functionNode::id().

Referenced by commonFuncsMgr::startCommonFuncs().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType srvAppLayer::DeleteFunction (BYTE *id*)

Method to delete function from service layer.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *id* | - [functionNode](#AAAAAAAAHI) identifier |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - function deleted successfully |

Definition at line 197 of file srvAppLayer.cpp.

References func\_quantity, and Functions.

Referenced by commonFuncsMgr::stopCommonFuncs().

Here is the caller graph for this function:

#### errType srvAppLayer::encodeBlock (rcsCmd \* *ss\_cmd*, BYTE \*\* *data*)

Method to encode data from rcsCmd message.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *ss\_cmd* | - rcsCmd message |
| out | *data* | - pointer to pointer that will include result of method |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - Block encoded successfully |

##### [Todo](#dd/da0/todo__todo000015):

nobody uses this method. need to be deleted.

Definition at line 214 of file srvAppLayer.cpp.

#### errType srvAppLayer::StartListening ()

Method to prepare and start base communication engine.

if errors has been occured,method initiate appTerminate signal.

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - communication engine starts successfully |
| *err\_sock\_error* | - udp sockets prerparing error |

1. Prepare [clientsRequestsQueue](#AAAAAAAAND) .

Queue stores all received messages from clients and clients adresses.

2. Prepare [functionsAnswersQueue](#AAAAAAAANE) .

Queue stores all functions answers to clients requests.

3. Start [udpListenerThread](#AAAAAAAAHT)

Thread listen for udp messages from clients and stores messages in [clientsRequestsQueue](#AAAAAAAAND) .

4. Prepare [equip\_listen](#AAAAAAAAMZ) .

This is an udp port instance that using to listen data from equipment

5. Start [udpSenderThread](#AAAAAAAAHS)

Thread sending udp messageds to clients from [functionsAnswersQueue](#AAAAAAAANE)

6. Start [equipListenPolling](#AAAAAAAAGG)

Thread listen for udp messages from equipment

Definition at line 229 of file srvAppLayer.cpp.

References AppTerminated, clientsRequestsQueue, eq\_udp\_listen\_port, equip\_listen, equipListenPolling(), equipListenThread, functionsAnswersQueue, ifCount, listenerThread, senderThread, udpListenerThread(), and udpSenderThread().

Referenced by appInit().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType srvAppLayer::StopListening ()

Method to stop base communication engine.

method initiate appTerminate signal, deletes queues.

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - engine stopped successfully |

Definition at line 274 of file srvAppLayer.cpp.

References AppTerminated, clientsRequestsQueue, functionsAnswersQueue, and verbose\_level.

Referenced by appDeinit().

Here is the caller graph for this function:

#### errType srvAppLayer::equip\_reading\_event ()

Method to asynchonous polling of [equip\_listen](#AAAAAAAAMZ) socket.

##### [Todo](#dd/da0/todo__todo000016):

too strange method.

May be it need be deleted.

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - udp socket received data |

Definition at line 301 of file srvAppLayer.cpp.

References equip\_listen.

Referenced by equipListenPolling().

Here is the caller graph for this function:

#### errType srvAppLayer::equip\_read\_data (BYTE \* *buffer*, size\_t \* *sz*)

Method to read data from [equip\_listen](#AAAAAAAAMZ) socket.

##### [Todo](#dd/da0/todo__todo000017):

too strange method.

May be it need be deleted.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| out | *buffer* | - uses to store recevied data |
| out | *sz* | - size in bytes of received data |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - udp socket received data has been readed |

Definition at line 316 of file srvAppLayer.cpp.

References equip\_listen.

Referenced by equipListenPolling().

Here is the caller graph for this function:

#### errType srvAppLayer::ProcessMessages ()

Method to make one step of [srvAppLayer](#AAAAAAAAGQ) step.

Method checks at start [AppTerminated](#AAAAAAAANH) signal, size of [clientsRequestsQueue](#AAAAAAAAND) and [rcvComplete\_flag](#AAAAAAAAHU) before it continuous execution.

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - one step of engine execution was successfully. |
| *err\_not\_init* | - method break execution. |

1) Read from [clientsRequestsQueue](#AAAAAAAAND) one new request

2) Decode readed request by [decodeMessage](#AAAAAAAANF)

3) Execute requested function if decoding was successfully by [execMessage](#AAAAAAAAJE)

4) Encoding function result ticket if execution was not successfully

5) Encode remains function results be [encodeFuncResult](#AAAAAAAAIY)

6) Write results to sending queue by [sendResult](#AAAAAAAANG)

7) Release allocated memory

8) Sync listening and sending threads by [rcvComplete\_flag](#AAAAAAAAHU) and [sndAllow\_flag](#AAAAAAAAHV)

Definition at line 488 of file srvAppLayer.cpp.

References AppTerminated, clientsRequestsQueue, decodeMessage(), encodeFuncResult(), execMessage(), Functions, ssBuffer::getFrontBlockSize(), ssBuffer::popBlock(), rcvComplete\_flag, sendResult(), ssBuffer::size(), and sndAllow\_flag.

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

#### WORD srvAppLayer::getListenerPortNum ()

getter for udp port number that listens all clients requests

Definition at line 565 of file srvAppLayer.cpp.

References cpListenerPortNum.

Referenced by udpListenerThread().

Here is the caller graph for this function:

#### BYTE srvAppLayer::terminated ()

Method to check application internal termination signal.

##### Return values:

|  |  |
| --- | --- |
| *0* | - Application runs normally |
| *1* | - Application terminating and preparing to exit to the OS |
| *2* | - Application terminating and preparing to reboot the OS |

Definition at line 148 of file srvAppLayer.cpp.

References AppTerminated.

Referenced by equipListenPolling(), main(), udpListenerThread(), and udpSenderThread().

Here is the caller graph for this function:

#### void srvAppLayer::terminate (BYTE *mode* = 1)

getter for AppTerminated signal

Definition at line 571 of file srvAppLayer.cpp.

References AppTerminated.

Referenced by udpListenerThread(), and udpSenderThread().

Here is the caller graph for this function:

#### [stateVector\_type](#AAAAAAAAIB) srvAppLayer::getStateVector ()

getter for ServiceState vector

Definition at line 577 of file srvAppLayer.cpp.

References ServiceState.

### Member Data Documentation

#### BYTE [srvAppLayer::AppTerminated](#AAAAAAAANH) [private]

Application termination process indicator

0 - Application run normally

1 - Applcation need to exit only

2 - Application need exit with reboot.

Definition at line 48 of file srvAppLayer.h.

Referenced by ProcessMessages(), srvAppLayer(), StartListening(), StopListening(), terminate(), and terminated().

#### [functionNode](#AAAAAAAAHI)\* [srvAppLayer::Functions](#AAAAAAAAJG)[100] [private]

Service functions array.

##### 100 functions at maximum. [Todo](#dd/da0/todo__todo000020):

array not the best data structure for this purposes.

Definition at line 53 of file srvAppLayer.h.

Referenced by CreateNewFunction(), decodeMessage(), DeleteFunction(), encodeFuncResult(), execMessage(), ProcessMessages(), and ~srvAppLayer().

#### BYTE [srvAppLayer::func\_quantity](#AAAAAAAANI) [private]

Counter that stores really declared functions quantity.

Definition at line 56 of file srvAppLayer.h.

Referenced by CreateNewFunction(), DeleteFunction(), srvAppLayer(), and ~srvAppLayer().

#### BYTE [srvAppLayer::ifCount](#AAAAAAAANJ) [private]

Counter of ethernet interfaces. No have an idea how to use it.

Definition at line 57 of file srvAppLayer.h.

Referenced by srvAppLayer(), and StartListening().

#### pthread\_t [srvAppLayer::listenerThread](#AAAAAAAANK) [private]

Handle to client requests listening thread.

Definition at line 59 of file srvAppLayer.h.

Referenced by StartListening().

#### pthread\_t [srvAppLayer::senderThread](#AAAAAAAANL) [private]

Handle to client answers sending thread.

Definition at line 60 of file srvAppLayer.h.

Referenced by StartListening().

#### pthread\_t [srvAppLayer::equipListenThread](#AAAAAAAANM) [private]

Handle to equipmnent data listening thread.

##### Note:

programm not have equipment data sending thread

Definition at line 61 of file srvAppLayer.h.

Referenced by StartListening().

#### WORD [srvAppLayer::cpListenerPortNum](#AAAAAAAANN) [private]

settings: Udp port number to listen requests from network clients

Definition at line 64 of file srvAppLayer.h.

Referenced by getListenerPortNum(), and srvAppLayer().

#### udp\_port\* [srvAppLayer::equip\_listen](#AAAAAAAAMZ) [private]

udp\_port instance that associates with listening data from equipment

Definition at line 65 of file srvAppLayer.h.

Referenced by equip\_read\_data(), equip\_reading\_event(), StartListening(), and ~srvAppLayer().

#### [stateVector\_type](#AAAAAAAAIB) [srvAppLayer::ServiceState](#AAAAAAAANO) [private]

Service state vector.

Definition at line 72 of file srvAppLayer.h.

Referenced by execMessage(), and getStateVector().

#### [ssBuffer](#AAAAAAAAES)\* [srvAppLayer::clientsRequestsQueue](#AAAAAAAAND)

Queue that stores received requests from client.

Definition at line 76 of file srvAppLayer.h.

Referenced by ProcessMessages(), StartListening(), StopListening(), and udpListenerThread().

#### [ssBuffer](#AAAAAAAAES)\* [srvAppLayer::functionsAnswersQueue](#AAAAAAAANE)

Queue that stores service functions answers to clients.

Definition at line 77 of file srvAppLayer.h.

Referenced by sendResult(), StartListening(), StopListening(), and udpSenderThread().

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

src/srvAppLayer/[srvAppLayer.h](#AAAAAAAAHZ)

src/srvAppLayer/[srvAppLayer.cpp](#AAAAAAAAHR)

## ssBlock Struct Reference

[ssBuffer](#AAAAAAAAES) list entry.

#include <ssBuffer.h>

### Public Attributes

sockaddr\_in [addr\_in](#AAAAAAAANP)

*recepient of datablock address,*

DWORD [dataLen](#AAAAAAAANQ)

*size in bytes of data,*

BYTE \* [data](#AAAAAAAANR)

*data array pointer.*

### Detailed Description

[ssBuffer](#AAAAAAAAES) list entry.

Definition at line 20 of file ssBuffer.h.

### Member Data Documentation

#### sockaddr\_in [ssBlock::addr\_in](#AAAAAAAANP)

recepient of datablock address,

Definition at line 22 of file ssBuffer.h.

Referenced by ssBuffer::dbgPrint(), ssBuffer::popBlock(), and ssBuffer::pushBlock().

#### DWORD [ssBlock::dataLen](#AAAAAAAANQ)

size in bytes of data,

Definition at line 23 of file ssBuffer.h.

Referenced by ssBuffer::dbgPrint(), ssBuffer::getFrontBlockSize(), ssBuffer::popBlock(), and ssBuffer::pushBlock().

#### BYTE\* [ssBlock::data](#AAAAAAAANR)

data array pointer.

Definition at line 24 of file ssBuffer.h.

Referenced by ssBuffer::dbgPrint(), ssBuffer::popBlock(), and ssBuffer::pushBlock().

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

src/buffer/[ssBuffer.h](#AAAAAAAAEU)

## ssBuffer Class Reference

list (deque) implementaion for storing [ssBlock](#AAAAAAAAEV) elements.

#include <ssBuffer.h>

Collaboration diagram for ssBuffer:

### Public Member Functions

[ssBuffer](#AAAAAAAANS) ()

[~ssBuffer](#AAAAAAAANT) ()

errType [pushBlock](#AAAAAAAANU) (sockaddr\_in \*, BYTE \*, DWORD len)

*push datablock to queue*

DWORD [popBlock](#AAAAAAAANV) (sockaddr\_in \*, BYTE \*)

*pop datablock from queue*

errType [getBlockPtrAt](#AAAAAAAANW) (int index, [ssBlock](#AAAAAAAAEV) \*block)

*read item from queue by* ***index***

DWORD [getFrontBlockSize](#AAAAAAAANX) ()

*get first in queue (front) block size in bytes*

DWORD [size](#AAAAAAAANY) ()

*return length (items quantity) of queue*

void [dbgPrint](#AAAAAAAANZ) ()

*Print for debug purposes contents of stored queue (in a hexadecimal notation)*

### Private Attributes

deque< [ssBlock](#AAAAAAAAEV) \* > [buffer](#AAAAAAAAOA)

### Detailed Description

list (deque) implementaion for storing [ssBlock](#AAAAAAAAEV) elements.

deque uses to organize sending or receiving queue

Definition at line 33 of file ssBuffer.h.

### Constructor & Destructor Documentation

#### ssBuffer::ssBuffer ()

Definition at line 23 of file ssBuffer.cpp.

#### ssBuffer::~ssBuffer ()

Definition at line 28 of file ssBuffer.cpp.

### Member Function Documentation

#### errType ssBuffer::pushBlock (sockaddr\_in \* *addr*, BYTE \* *block*, DWORD *len*)

push datablock to queue

copies data from **block** and **addr** pointers to new memory locations

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *addr* | - recepient address (owner of datablock) |
| in | *block* | - datablock |
| in | *len* | - size in bytes of datablock |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - pushing was successfully |

Definition at line 41 of file ssBuffer.cpp.

References ssBlock::addr\_in, ssBlock::data, and ssBlock::dataLen.

Referenced by deqUdp::readData(), and srvAppLayer::sendResult().

Here is the caller graph for this function:

#### DWORD ssBuffer::popBlock (sockaddr\_in \* *addr*, BYTE \* *block*)

pop datablock from queue

copies data from queue to **block** and **addr** pointers

**block** and **addr** need to be allocated before calling this method.

queue element will be removed from queue

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *addr* | - recepient address (owner of datablock) |
| in | *block* | - datablock |

##### Return values:

|  |  |
| --- | --- |
| *lenght* | - size in bytes of readed datablock |

Definition at line 77 of file ssBuffer.cpp.

References ssBlock::addr\_in, ssBlock::data, and ssBlock::dataLen.

Referenced by srvAppLayer::ProcessMessages(), and deqUdp::sendData().

Here is the caller graph for this function:

#### errType ssBuffer::getBlockPtrAt (int *index*, [ssBlock](#AAAAAAAAEV) \* *block*)

read item from queue by **index**

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *index* | - queue item index |
| out | *block* | - readed queue item |

Definition at line 99 of file ssBuffer.cpp.

#### DWORD ssBuffer::getFrontBlockSize ()

get first in queue (front) block size in bytes

need to be prepared for reading next queue element

##### Return values:

|  |  |
| --- | --- |
| *dataLen* | - size in bytes of data block |

Definition at line 61 of file ssBuffer.cpp.

References ssBlock::dataLen.

Referenced by srvAppLayer::ProcessMessages(), and deqUdp::sendData().

Here is the caller graph for this function:

#### DWORD ssBuffer::size ()

return length (items quantity) of queue

Definition at line 112 of file ssBuffer.cpp.

Referenced by dbgPrint(), and srvAppLayer::ProcessMessages().

Here is the caller graph for this function:

#### void ssBuffer::dbgPrint ()

Print for debug purposes contents of stored queue (in a hexadecimal notation)

Definition at line 123 of file ssBuffer.cpp.

References ssBlock::addr\_in, ssBlock::data, ssBlock::dataLen, and size().

Here is the call graph for this function:

### Member Data Documentation

#### deque<[ssBlock](#AAAAAAAAEV)\*> [ssBuffer::buffer](#AAAAAAAAOA) [private]

Definition at line 35 of file ssBuffer.h.

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

src/buffer/[ssBuffer.h](#AAAAAAAAEU)

src/buffer/[ssBuffer.cpp](#AAAAAAAAER)

## stateVector\_type Struct Reference

Main vector of service base states.

#include <srvAppLayer.h>

Collaboration diagram for stateVector\_type:

### Public Attributes

BYTE [reserved](#AAAAAAAAOB)

[serviceState](#AAAAAAAAIA) [state](#AAAAAAAAOC)

errType [lastResult](#AAAAAAAAOD)

BYTE [lastFuncId](#AAAAAAAAOE)

### Detailed Description

Main vector of service base states.

Definition at line 34 of file srvAppLayer.h.

### Member Data Documentation

#### BYTE [stateVector\_type::reserved](#AAAAAAAAOB)

Definition at line 35 of file srvAppLayer.h.

#### [serviceState](#AAAAAAAAIA) [stateVector\_type::state](#AAAAAAAAOC)

Definition at line 36 of file srvAppLayer.h.

#### errType [stateVector\_type::lastResult](#AAAAAAAAOD)

Definition at line 37 of file srvAppLayer.h.

Referenced by srvAppLayer::execMessage().

#### BYTE [stateVector\_type::lastFuncId](#AAAAAAAAOE)

Definition at line 38 of file srvAppLayer.h.

Referenced by srvAppLayer::execMessage().

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

src/srvAppLayer/[srvAppLayer.h](#AAAAAAAAHZ)

# File Documentation

## src/\_auto\_config.h File Reference

### Defines

#define [HAVE\_ARPA\_INET\_H](#AAAAAAAAAC)  1

#define [HAVE\_INTTYPES\_H](#AAAAAAAAAD)  1

#define [HAVE\_MALLOC](#AAAAAAAAAE)  1

#define [HAVE\_MEMORY\_H](#AAAAAAAAAF)  1

#define [HAVE\_NETINET\_IN\_H](#AAAAAAAAAG)  1

#define [HAVE\_REALLOC](#AAAAAAAAAH)  1

#define [HAVE\_SOCKET](#AAAAAAAAAI)  1

#define [HAVE\_STDBOOL\_H](#AAAAAAAAAJ)  1

#define [HAVE\_STDDEF\_H](#AAAAAAAAAK)  1

#define [HAVE\_STDINT\_H](#AAAAAAAAAL)  1

#define [HAVE\_STDLIB\_H](#AAAAAAAAAM)  1

#define [HAVE\_STRINGS\_H](#AAAAAAAAAN)  1

#define [HAVE\_STRING\_H](#AAAAAAAAAO)  1

#define [HAVE\_SYS\_SOCKET\_H](#AAAAAAAAAP)  1

#define [HAVE\_SYS\_STAT\_H](#AAAAAAAAAQ)  1

#define [HAVE\_SYS\_TYPES\_H](#AAAAAAAAAR)  1

#define [HAVE\_UNISTD\_H](#AAAAAAAAAS)  1

#define [HAVE\_\_BOOL](#AAAAAAAAAT)  1

#define [PACKAGE](#AAAAAAAAAU)  "r168"

#define [PACKAGE\_BUGREPORT](#AAAAAAAAAV)  "nosenko@rec-etu.com"

#define [PACKAGE\_NAME](#AAAAAAAAAW)  "src/main.cpp"

#define [PACKAGE\_STRING](#AAAAAAAAAX)  "src/main.cpp 0.1"

#define [PACKAGE\_TARNAME](#AAAAAAAAAY)  "src-main-cpp"

#define [PACKAGE\_URL](#AAAAAAAAAZ)  ""

#define [PACKAGE\_VERSION](#AAAAAAAABA)  "0.1"

#define [STDC\_HEADERS](#AAAAAAAABB)  1

#define [VERSION](#AAAAAAAABC)  "0.1"

### Define Documentation

#### #define HAVE\_ARPA\_INET\_H  1

Definition at line 5 of file \_auto\_config.h.

#### #define HAVE\_INTTYPES\_H  1

Definition at line 8 of file \_auto\_config.h.

#### #define HAVE\_MALLOC  1

Definition at line 12 of file \_auto\_config.h.

#### #define HAVE\_MEMORY\_H  1

Definition at line 15 of file \_auto\_config.h.

#### #define HAVE\_NETINET\_IN\_H  1

Definition at line 18 of file \_auto\_config.h.

#### #define HAVE\_REALLOC  1

Definition at line 22 of file \_auto\_config.h.

#### #define HAVE\_SOCKET  1

Definition at line 25 of file \_auto\_config.h.

#### #define HAVE\_STDBOOL\_H  1

Definition at line 28 of file \_auto\_config.h.

#### #define HAVE\_STDDEF\_H  1

Definition at line 31 of file \_auto\_config.h.

#### #define HAVE\_STDINT\_H  1

Definition at line 34 of file \_auto\_config.h.

#### #define HAVE\_STDLIB\_H  1

Definition at line 37 of file \_auto\_config.h.

#### #define HAVE\_STRINGS\_H  1

Definition at line 40 of file \_auto\_config.h.

#### #define HAVE\_STRING\_H  1

Definition at line 43 of file \_auto\_config.h.

#### #define HAVE\_SYS\_SOCKET\_H  1

Definition at line 46 of file \_auto\_config.h.

#### #define HAVE\_SYS\_STAT\_H  1

Definition at line 49 of file \_auto\_config.h.

#### #define HAVE\_SYS\_TYPES\_H  1

Definition at line 52 of file \_auto\_config.h.

#### #define HAVE\_UNISTD\_H  1

Definition at line 55 of file \_auto\_config.h.

#### #define HAVE\_\_BOOL  1

Definition at line 58 of file \_auto\_config.h.

#### #define PACKAGE  "r168"

Definition at line 61 of file \_auto\_config.h.

#### #define PACKAGE\_BUGREPORT  "nosenko@rec-etu.com"

Definition at line 64 of file \_auto\_config.h.

#### #define PACKAGE\_NAME  "src/main.cpp"

Definition at line 67 of file \_auto\_config.h.

#### #define PACKAGE\_STRING  "src/main.cpp 0.1"

Definition at line 70 of file \_auto\_config.h.

#### #define PACKAGE\_TARNAME  "src-main-cpp"

Definition at line 73 of file \_auto\_config.h.

#### #define PACKAGE\_URL  ""

Definition at line 76 of file \_auto\_config.h.

#### #define PACKAGE\_VERSION  "0.1"

Definition at line 79 of file \_auto\_config.h.

#### #define STDC\_HEADERS  1

Definition at line 82 of file \_auto\_config.h.

#### #define VERSION  "0.1"

Definition at line 85 of file \_auto\_config.h.

## src/\_global.cpp File Reference

Global environment.

#include <netinet/in.h>

#include <queue>

#include "../../rcsLib/ortsTypes/ortsTypes.h"

#include "buffer/ssBuffer.h"

#include "../../udp\_port/udp\_port.h"

#include "../../rcsLib/rcsCmd/rcsCmd.h"

#include "ICAppLayer/FunctionNode/param\_desc.h"

#include "ICAppLayer/FunctionNode/FunctionNode.h"

#include "ICAppLayer/ICAppLayer.h"

### Variables

int [verbose\_level](#AAAAAAAABE) = 0

*Debug detail level printing.*

bool [listen\_mode](#AAAAAAAABF) = false

*needless variable.*

bool [awaitingPattern\_mode](#AAAAAAAABG) = false

*needless variable.*

bool [pattern\_found](#AAAAAAAABH) = false

*needless variable.*

char [patternFile](#AAAAAAAABI) [255] = {0}

*needless variable.*

char [reactionFile](#AAAAAAAABJ) [255] = {0}

*needless variable.*

char [dataFile](#AAAAAAAABK) [255] = {0}

*needless variable.*

char [if\_name](#AAAAAAAABL) [255] = {0}

*needless variable.*

bool [AppTerminated](#AAAAAAAABM) = false

*Programm termination signal.*

char [eq\_ip\_addr](#AAAAAAAABN) [255] = {0}

*Equipment IP address.*

WORD [wUdp](#AAAAAAAABO) = 0

*Server udp port number for communicate with client.*

WORD [eq\_udp\_listen\_port](#AAAAAAAABP) = 0

*Server udp port number for listen an equipment.*

WORD [eq\_udp\_sending\_port](#AAAAAAAABQ) = 0

*Server udp port number for sending into equipment.*

in\_addr [equipAddr](#AAAAAAAABR)

*Storage for in\_addr of equipment.*

### Detailed Description

Global environment.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [\_global.cpp](#AAAAAAAABS).

### Variable Documentation

#### int [verbose\_level](#AAAAAAAABT) = 0

Debug detail level printing.

Definition at line 26 of file \_global.cpp.

Referenced by appInit(), process\_cmdLine(), srvAppLayer::StopListening(), and term\_handler().

#### bool [listen\_mode](#AAAAAAAABU) = false

needless variable.

##### [Todo](#dd/da0/todo__todo000001):

Need to be deleted.

Definition at line 29 of file \_global.cpp.

#### bool [awaitingPattern\_mode](#AAAAAAAABV) = false

needless variable.

##### [Todo](#dd/da0/todo__todo000002):

Need to be deleted.

Definition at line 30 of file \_global.cpp.

#### bool [pattern\_found](#AAAAAAAABW) = false

needless variable.

##### [Todo](#dd/da0/todo__todo000003):

Need to be deleted.

Definition at line 31 of file \_global.cpp.

#### char [patternFile](#AAAAAAAABX)[255] = {0}

needless variable.

##### [Todo](#dd/da0/todo__todo000004):

Need to be deleted.

Definition at line 32 of file \_global.cpp.

#### char [reactionFile](#AAAAAAAABY)[255] = {0}

needless variable.

##### [Todo](#dd/da0/todo__todo000005):

Need to be deleted.

Definition at line 33 of file \_global.cpp.

#### char [dataFile](#AAAAAAAABZ)[255] = {0}

needless variable.

##### [Todo](#dd/da0/todo__todo000006):

Need to be deleted.

Definition at line 34 of file \_global.cpp.

#### char [if\_name](#AAAAAAAACA)[255] = {0}

needless variable.

##### [Todo](#dd/da0/todo__todo000007):

Need to be deleted.

Definition at line 35 of file \_global.cpp.

#### bool [AppTerminated](#AAAAAAAACB) = false

Programm termination signal.

All processes need to finish own job.

Definition at line 38 of file \_global.cpp.

Referenced by main().

#### char [eq\_ip\_addr](#AAAAAAAACC)[255] = {0}

Equipment IP address.

Definition at line 40 of file \_global.cpp.

Referenced by process\_cmdLine().

#### WORD [wUdp](#AAAAAAAACD) = 0

Server udp port number for communicate with client.

Definition at line 43 of file \_global.cpp.

Referenced by main(), process\_cmdLine(), srvAppLayer::sendResult(), udpListenerThread(), and udpSenderThread().

#### WORD [eq\_udp\_listen\_port](#AAAAAAAACE) = 0

Server udp port number for listen an equipment.

Definition at line 44 of file \_global.cpp.

Referenced by main(), process\_cmdLine(), and srvAppLayer::StartListening().

#### WORD [eq\_udp\_sending\_port](#AAAAAAAACF) = 0

Server udp port number for sending into equipment.

Definition at line 45 of file \_global.cpp.

Referenced by main(), and process\_cmdLine().

#### in\_addr [equipAddr](#AAAAAAAACG)

Storage for in\_addr of equipment.

Definition at line 47 of file \_global.cpp.

Referenced by main(), and process\_cmdLine().

## src/\_global.h File Reference

Global environment interface header.

### Variables

bool [AppTerminated](#AAAAAAAACB)

*Programm termination signal.*

bool [awaitingPattern\_mode](#AAAAAAAABV)

*needless variable.*

bool [pattern\_found](#AAAAAAAABW)

*needless variable.*

int [verbose\_level](#AAAAAAAABT)

*Debug detail level printing.*

bool [listen\_mode](#AAAAAAAABU)

*needless variable.*

char [eq\_ip\_addr](#AAAAAAAACC) [255]

*Equipment IP address.*

char [dataFile](#AAAAAAAABZ) [255]

*needless variable.*

char [if\_name](#AAAAAAAACA) [255]

*needless variable.*

char [patternFile](#AAAAAAAABX) [255]

*needless variable.*

char [reactionFile](#AAAAAAAABY) [255]

*needless variable.*

WORD [eq\_udp\_listen\_port](#AAAAAAAACE)

*Server udp port number for listen an equipment.*

WORD [eq\_udp\_sending\_port](#AAAAAAAACF)

*Server udp port number for sending into equipment.*

in\_addr [equipAddr](#AAAAAAAACG)

*Storage for in\_addr of equipment.*

WORD [wUdp](#AAAAAAAACD)

*Server udp port number for communicate with client.*

### Detailed Description

Global environment interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [\_global.h](#AAAAAAAACI).

### Variable Documentation

#### bool [AppTerminated](#AAAAAAAACB)

Programm termination signal.

All processes need to finish own job.

Definition at line 38 of file \_global.cpp.

Referenced by main().

#### bool [awaitingPattern\_mode](#AAAAAAAABV)

needless variable.

##### [Todo](#dd/da0/todo__todo000002):

Need to be deleted.

Definition at line 30 of file \_global.cpp.

#### bool [pattern\_found](#AAAAAAAABW)

needless variable.

##### [Todo](#dd/da0/todo__todo000003):

Need to be deleted.

Definition at line 31 of file \_global.cpp.

#### int [verbose\_level](#AAAAAAAABT)

Debug detail level printing.

Definition at line 26 of file \_global.cpp.

Referenced by appInit(), process\_cmdLine(), srvAppLayer::StopListening(), and term\_handler().

#### bool [listen\_mode](#AAAAAAAABU)

needless variable.

##### [Todo](#dd/da0/todo__todo000001):

Need to be deleted.

Definition at line 29 of file \_global.cpp.

#### char [eq\_ip\_addr](#AAAAAAAACC)[255]

Equipment IP address.

Definition at line 40 of file \_global.cpp.

Referenced by process\_cmdLine().

#### char [dataFile](#AAAAAAAABZ)[255]

needless variable.

##### [Todo](#dd/da0/todo__todo000006):

Need to be deleted.

Definition at line 34 of file \_global.cpp.

#### char [if\_name](#AAAAAAAACA)[255]

needless variable.

##### [Todo](#dd/da0/todo__todo000007):

Need to be deleted.

Definition at line 35 of file \_global.cpp.

#### char [patternFile](#AAAAAAAABX)[255]

needless variable.

##### [Todo](#dd/da0/todo__todo000004):

Need to be deleted.

Definition at line 32 of file \_global.cpp.

#### char [reactionFile](#AAAAAAAABY)[255]

needless variable.

##### [Todo](#dd/da0/todo__todo000005):

Need to be deleted.

Definition at line 33 of file \_global.cpp.

#### WORD [eq\_udp\_listen\_port](#AAAAAAAACE)

Server udp port number for listen an equipment.

Definition at line 44 of file \_global.cpp.

Referenced by main(), process\_cmdLine(), and srvAppLayer::StartListening().

#### WORD [eq\_udp\_sending\_port](#AAAAAAAACF)

Server udp port number for sending into equipment.

Definition at line 45 of file \_global.cpp.

Referenced by main(), and process\_cmdLine().

#### in\_addr [equipAddr](#AAAAAAAACG)

Storage for in\_addr of equipment.

Definition at line 47 of file \_global.cpp.

Referenced by main(), and process\_cmdLine().

#### WORD [wUdp](#AAAAAAAACD)

Server udp port number for communicate with client.

Definition at line 43 of file \_global.cpp.

Referenced by main(), process\_cmdLine(), srvAppLayer::sendResult(), udpListenerThread(), and udpSenderThread().

## src/arg\_parser/carg\_parser.cpp File Reference

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include "carg\_parser.h"

### Functions

char [ap\_resize\_buffer](#AAAAAAAACK) (void \*buf, const int min\_size)

char [push\_back\_record](#AAAAAAAACL) ([Arg\_parser](#AAAAAAAACM) \*ap, const int code, const char \*argument)

char [add\_error](#AAAAAAAACN) ([Arg\_parser](#AAAAAAAACM) \*ap, const char \*msg)

void [free\_data](#AAAAAAAACO) ([Arg\_parser](#AAAAAAAACM) \*ap)

char [parse\_long\_option](#AAAAAAAACP) ([Arg\_parser](#AAAAAAAACM) \*ap, const char \*const opt, const char \*const arg, const [ap\_Option](#AAAAAAAACQ) options[], int \*argindp)

char [parse\_short\_option](#AAAAAAAACR) ([Arg\_parser](#AAAAAAAACM) \*ap, const char \*const opt, const char \*const arg, const [ap\_Option](#AAAAAAAACQ) options[], int \*argindp)

char [ap\_init](#AAAAAAAACS) ([Arg\_parser](#AAAAAAAACM) \*ap, const int argc, const char \*const argv[], const [ap\_Option](#AAAAAAAACQ) options[], const char in\_order)

void [ap\_free](#AAAAAAAACT) ([Arg\_parser](#AAAAAAAACM) \*ap)

const char \* [ap\_error](#AAAAAAAACU) (const [Arg\_parser](#AAAAAAAACM) \*ap)

int [ap\_arguments](#AAAAAAAACV) (const [Arg\_parser](#AAAAAAAACM) \*ap)

int [ap\_code](#AAAAAAAACW) (const [Arg\_parser](#AAAAAAAACM) \*ap, const int i)

const char \* [ap\_argument](#AAAAAAAACX) (const [Arg\_parser](#AAAAAAAACM) \*ap, const int i)

### Function Documentation

#### char ap\_resize\_buffer (void \* *buf*, const int *min\_size*)

Definition at line 26 of file carg\_parser.cpp.

Referenced by add\_error(), ap\_init(), and push\_back\_record().

Here is the caller graph for this function:

#### char push\_back\_record ([Arg\_parser](#AAAAAAAACM) \* *ap*, const int *code*, const char \* *argument*)

Definition at line 37 of file carg\_parser.cpp.

References ap\_resize\_buffer(), ap\_Record::argument, ap\_Record::code, Arg\_parser::data, and Arg\_parser::data\_size.

Referenced by ap\_init(), parse\_long\_option(), and parse\_short\_option().

Here is the call graph for this function:

Here is the caller graph for this function:

#### char add\_error ([Arg\_parser](#AAAAAAAACM) \* *ap*, const char \* *msg*)

Definition at line 54 of file carg\_parser.cpp.

References ap\_resize\_buffer(), Arg\_parser::error, and Arg\_parser::error\_size.

Referenced by parse\_long\_option(), and parse\_short\_option().

Here is the call graph for this function:

Here is the caller graph for this function:

#### void free\_data ([Arg\_parser](#AAAAAAAACM) \* *ap*)

Definition at line 65 of file carg\_parser.cpp.

References ap\_Record::argument, Arg\_parser::data, and Arg\_parser::data\_size.

Referenced by ap\_free(), and ap\_init().

Here is the caller graph for this function:

#### char parse\_long\_option ([Arg\_parser](#AAAAAAAACM) \* *ap*, const char \*const *opt*, const char \*const *arg*, const [ap\_Option](#AAAAAAAACQ) *options*[], int \* *argindp*)

Definition at line 74 of file carg\_parser.cpp.

References add\_error(), ap\_no, ap\_yes, ap\_Option::code, and push\_back\_record().

Referenced by ap\_init().

Here is the call graph for this function:

Here is the caller graph for this function:

#### char parse\_short\_option ([Arg\_parser](#AAAAAAAACM) \* *ap*, const char \*const *opt*, const char \*const *arg*, const [ap\_Option](#AAAAAAAACQ) *options*[], int \* *argindp*)

Definition at line 146 of file carg\_parser.cpp.

References add\_error(), ap\_no, ap\_yes, ap\_Option::code, and push\_back\_record().

Referenced by ap\_init().

Here is the call graph for this function:

Here is the caller graph for this function:

#### char ap\_init ([Arg\_parser](#AAAAAAAACM) \* *ap*, const int *argc*, const char \*const *argv*[], const [ap\_Option](#AAAAAAAACQ) *options*[], const char *in\_order*)

Definition at line 194 of file carg\_parser.cpp.

References ap\_resize\_buffer(), Arg\_parser::data, Arg\_parser::data\_size, Arg\_parser::error, Arg\_parser::error\_size, free\_data(), parse\_long\_option(), parse\_short\_option(), and push\_back\_record().

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

#### void ap\_free ([Arg\_parser](#AAAAAAAACM) \* *ap*)

Definition at line 253 of file carg\_parser.cpp.

References Arg\_parser::error, Arg\_parser::error\_size, and free\_data().

Here is the call graph for this function:

#### const char\* ap\_error (const [Arg\_parser](#AAAAAAAACM) \* *ap*)

Definition at line 261 of file carg\_parser.cpp.

References Arg\_parser::error.

Referenced by process\_cmdLine().

Here is the caller graph for this function:

#### int ap\_arguments (const [Arg\_parser](#AAAAAAAACM) \* *ap*)

Definition at line 264 of file carg\_parser.cpp.

References Arg\_parser::data\_size.

Referenced by ap\_argument(), ap\_code(), and process\_cmdLine().

Here is the caller graph for this function:

#### int ap\_code (const [Arg\_parser](#AAAAAAAACM) \* *ap*, const int *i*)

Definition at line 267 of file carg\_parser.cpp.

References ap\_arguments(), ap\_Record::code, and Arg\_parser::data.

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

#### const char\* ap\_argument (const [Arg\_parser](#AAAAAAAACM) \* *ap*, const int *i*)

Definition at line 274 of file carg\_parser.cpp.

References ap\_arguments(), ap\_Record::argument, and Arg\_parser::data.

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

## src/arg\_parser/carg\_parser.h File Reference

### Classes

struct [ap\_Option](#AAAAAAAACQ)

struct [ap\_Record](#AAAAAAAACZ)

struct [Arg\_parser](#AAAAAAAACM)

### Enumerations

enum [ap\_Has\_arg](#AAAAAAAADA) { [ap\_no](#AAAAAAAADB), [ap\_yes](#AAAAAAAADC), [ap\_maybe](#AAAAAAAADD) }

### Functions

char [ap\_init](#AAAAAAAADE) ([Arg\_parser](#AAAAAAAACM) \*ap, const int argc, const char \*const argv[], const [ap\_Option](#AAAAAAAACQ) options[], const char in\_order)

void [ap\_free](#AAAAAAAADF) ([Arg\_parser](#AAAAAAAACM) \*ap)

const char \* [ap\_error](#AAAAAAAADG) (const [Arg\_parser](#AAAAAAAACM) \*ap)

int [ap\_arguments](#AAAAAAAADH) (const [Arg\_parser](#AAAAAAAACM) \*ap)

int [ap\_code](#AAAAAAAADI) (const [Arg\_parser](#AAAAAAAACM) \*ap, const int i)

const char \* [ap\_argument](#AAAAAAAADJ) (const [Arg\_parser](#AAAAAAAACM) \*ap, const int i)

### Enumeration Type Documentation

#### enum [ap\_Has\_arg](#AAAAAAAADA)

**Enumerator:**

***ap\_no***

***ap\_yes***

***ap\_maybe***

Definition at line 42 of file carg\_parser.h.

### Function Documentation

#### char ap\_init ([Arg\_parser](#AAAAAAAACM) \* *ap*, const int *argc*, const char \*const *argv*[], const [ap\_Option](#AAAAAAAACQ) *options*[], const char *in\_order*)

Definition at line 194 of file carg\_parser.cpp.

References ap\_resize\_buffer(), Arg\_parser::data, Arg\_parser::data\_size, Arg\_parser::error, Arg\_parser::error\_size, free\_data(), parse\_long\_option(), parse\_short\_option(), and push\_back\_record().

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

#### void ap\_free ([Arg\_parser](#AAAAAAAACM) \* *ap*)

Definition at line 253 of file carg\_parser.cpp.

References Arg\_parser::error, Arg\_parser::error\_size, and free\_data().

Here is the call graph for this function:

#### const char\* ap\_error (const [Arg\_parser](#AAAAAAAACM) \* *ap*)

Definition at line 261 of file carg\_parser.cpp.

References Arg\_parser::error.

Referenced by process\_cmdLine().

Here is the caller graph for this function:

#### int ap\_arguments (const [Arg\_parser](#AAAAAAAACM) \* *ap*)

Definition at line 264 of file carg\_parser.cpp.

References Arg\_parser::data\_size.

Referenced by ap\_argument(), ap\_code(), and process\_cmdLine().

Here is the caller graph for this function:

#### int ap\_code (const [Arg\_parser](#AAAAAAAACM) \* *ap*, const int *i*)

Definition at line 267 of file carg\_parser.cpp.

References ap\_arguments(), ap\_Record::code, and Arg\_parser::data.

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

#### const char\* ap\_argument (const [Arg\_parser](#AAAAAAAACM) \* *ap*, const int *i*)

Definition at line 274 of file carg\_parser.cpp.

References ap\_arguments(), ap\_Record::argument, and Arg\_parser::data.

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

## src/auto\_config.h File Reference

### Defines

#define [HAVE\_ARPA\_INET\_H](#AAAAAAAADL)  1

#define [HAVE\_INTTYPES\_H](#AAAAAAAADM)  1

#define [HAVE\_MALLOC](#AAAAAAAADN)  1

#define [HAVE\_MEMORY\_H](#AAAAAAAADO)  1

#define [HAVE\_NETINET\_IN\_H](#AAAAAAAADP)  1

#define [HAVE\_REALLOC](#AAAAAAAADQ)  1

#define [HAVE\_SOCKET](#AAAAAAAADR)  1

#define [HAVE\_STDBOOL\_H](#AAAAAAAADS)  1

#define [HAVE\_STDDEF\_H](#AAAAAAAADT)  1

#define [HAVE\_STDINT\_H](#AAAAAAAADU)  1

#define [HAVE\_STDLIB\_H](#AAAAAAAADV)  1

#define [HAVE\_STRINGS\_H](#AAAAAAAADW)  1

#define [HAVE\_STRING\_H](#AAAAAAAADX)  1

#define [HAVE\_SYS\_SOCKET\_H](#AAAAAAAADY)  1

#define [HAVE\_SYS\_STAT\_H](#AAAAAAAADZ)  1

#define [HAVE\_SYS\_TYPES\_H](#AAAAAAAAEA)  1

#define [HAVE\_UNISTD\_H](#AAAAAAAAEB)  1

#define [HAVE\_\_BOOL](#AAAAAAAAEC)  1

#define [PACKAGE](#AAAAAAAAED)  "r168"

#define [PACKAGE\_BUGREPORT](#AAAAAAAAEE)  "nosenko@rec-etu.com"

#define [PACKAGE\_NAME](#AAAAAAAAEF)  "src/main.cpp"

#define [PACKAGE\_STRING](#AAAAAAAAEG)  "src/main.cpp 0.1"

#define [PACKAGE\_TARNAME](#AAAAAAAAEH)  "src-main-cpp"

#define [PACKAGE\_URL](#AAAAAAAAEI)  ""

#define [PACKAGE\_VERSION](#AAAAAAAAEJ)  "0.1"

#define [STDC\_HEADERS](#AAAAAAAAEK)  1

#define [VERSION](#AAAAAAAAEL)  "0.1"

### Define Documentation

#### #define HAVE\_ARPA\_INET\_H  1

Definition at line 5 of file auto\_config.h.

#### #define HAVE\_INTTYPES\_H  1

Definition at line 8 of file auto\_config.h.

#### #define HAVE\_MALLOC  1

Definition at line 12 of file auto\_config.h.

#### #define HAVE\_MEMORY\_H  1

Definition at line 15 of file auto\_config.h.

#### #define HAVE\_NETINET\_IN\_H  1

Definition at line 18 of file auto\_config.h.

#### #define HAVE\_REALLOC  1

Definition at line 22 of file auto\_config.h.

#### #define HAVE\_SOCKET  1

Definition at line 25 of file auto\_config.h.

#### #define HAVE\_STDBOOL\_H  1

Definition at line 28 of file auto\_config.h.

#### #define HAVE\_STDDEF\_H  1

Definition at line 31 of file auto\_config.h.

#### #define HAVE\_STDINT\_H  1

Definition at line 34 of file auto\_config.h.

#### #define HAVE\_STDLIB\_H  1

Definition at line 37 of file auto\_config.h.

#### #define HAVE\_STRINGS\_H  1

Definition at line 40 of file auto\_config.h.

#### #define HAVE\_STRING\_H  1

Definition at line 43 of file auto\_config.h.

#### #define HAVE\_SYS\_SOCKET\_H  1

Definition at line 46 of file auto\_config.h.

#### #define HAVE\_SYS\_STAT\_H  1

Definition at line 49 of file auto\_config.h.

#### #define HAVE\_SYS\_TYPES\_H  1

Definition at line 52 of file auto\_config.h.

#### #define HAVE\_UNISTD\_H  1

Definition at line 55 of file auto\_config.h.

#### #define HAVE\_\_BOOL  1

Definition at line 58 of file auto\_config.h.

#### #define PACKAGE  "r168"

Definition at line 61 of file auto\_config.h.

#### #define PACKAGE\_BUGREPORT  "nosenko@rec-etu.com"

Definition at line 64 of file auto\_config.h.

#### #define PACKAGE\_NAME  "src/main.cpp"

Definition at line 67 of file auto\_config.h.

#### #define PACKAGE\_STRING  "src/main.cpp 0.1"

Definition at line 70 of file auto\_config.h.

#### #define PACKAGE\_TARNAME  "src-main-cpp"

Definition at line 73 of file auto\_config.h.

#### #define PACKAGE\_URL  ""

Definition at line 76 of file auto\_config.h.

#### #define PACKAGE\_VERSION  "0.1"

Definition at line 79 of file auto\_config.h.

#### #define STDC\_HEADERS  1

Definition at line 82 of file auto\_config.h.

#### #define VERSION  "0.1"

Definition at line 85 of file auto\_config.h.

## src/buffer/buffer.cpp File Reference

Class [buffer](#AAAAAAAAEN) implementation.

#include <string.h>

#include <stdio.h>

#include "../../../rcsLib/ortsTypes/ortsTypes.h"

#include "buffer.h"

### Detailed Description

Class [buffer](#AAAAAAAAEN) implementation.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [buffer.cpp](#AAAAAAAAEO).

## src/buffer/buffer.h File Reference

Class [buffer](#AAAAAAAAEN) interface header.

### Classes

class [buffer](#AAAAAAAAEN)

*Simple queue of bytes.*

### Detailed Description

Class [buffer](#AAAAAAAAEN) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [buffer.h](#AAAAAAAAEQ).

## src/buffer/ssBuffer.cpp File Reference

Class [ssBuffer](#AAAAAAAAES) implementation.

#include <string.h>

#include <stdio.h>

#include <deque>

#include <netinet/in.h>

#include <arpa/inet.h>

#include "../../../rcsLib/ortsTypes/ortsTypes.h"

#include "ssBuffer.h"

### Detailed Description

Class [ssBuffer](#AAAAAAAAES) implementation.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [ssBuffer.cpp](#AAAAAAAAET).

## src/buffer/ssBuffer.h File Reference

Class [ssBuffer](#AAAAAAAAES) interface header.

### Classes

struct [ssBlock](#AAAAAAAAEV)

[*ssBuffer*](#AAAAAAAAES) *list entry.* class [ssBuffer](#AAAAAAAAES)

### *list (deque) implementaion for storing* [*ssBlock*](#AAAAAAAAEV) *elements.* Typedefs

typedef struct [ssBlock](#AAAAAAAAEV) [ssBlock](#AAAAAAAAEW)

### Detailed Description

Class [ssBuffer](#AAAAAAAAES) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [ssBuffer.h](#AAAAAAAAEX).

### Typedef Documentation

#### typedef struct [ssBlock](#AAAAAAAAEV) [ssBlock](#AAAAAAAAEV)

#### 

## src/config.h File Reference

## src/console\_out.cpp File Reference

aided functions to process\_cmdLine

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "auto\_config.h"

#include "./arg\_parser/carg\_parser.h"

### Defines

#define [REVISION](#AAAAAAAAFA)  0

*programm revision number*

### Functions

void [show\_help](#AAAAAAAAFB) (const char verbose)

*Show cmdline help information (--help)*

void [show\_version](#AAAAAAAAFC) ()

*Show programm version information (--version)*

void [show\_error](#AAAAAAAAFD) (const char \*msg, const int errcode, const char help)

*Show cmdLine parser error.*

void [internal\_error](#AAAAAAAAFE) (const char \*msg)

*Show cmdLine internal error.*

const char \* [optname](#AAAAAAAAFF) (const int code, const [ap\_Option](#AAAAAAAACQ) options[])

*Convert code with option from cmdLine argument to char buffer.*

### Variables

char [PROGVERSION](#AAAAAAAAFG) [255] = "0.1"

*version of programm*

char [Program\_name](#AAAAAAAAFH) [255] = "Ð¡ÐµÑ€Ð²Ð¸Ñ ÑÐ»ÑƒÐ¶ÐµÐ±Ð½Ð¾Ð¹ ÑÐ¸ÑÑ‚ÐµÐ¼Ñ‹"

*name of programm*

char [program\_name](#AAAAAAAAFI) [255] = "ss\_Service"

*filename of programm*

char [program\_year](#AAAAAAAAFJ) [255] = "2010"

*copyright year*

### Detailed Description

aided functions to process\_cmdLine

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [console\_out.cpp](#AAAAAAAAFK).

### Define Documentation

#### #define REVISION  0

programm revision number

revision by default is 0

Definition at line 24 of file console\_out.cpp.

Referenced by show\_version().

### Function Documentation

#### void show\_help (const char *verbose*)

Show cmdline help information (--help)

Definition at line 36 of file console\_out.cpp.

References program\_name, and Program\_name.

Referenced by process\_cmdLine().

Here is the caller graph for this function:

#### void show\_version ()

Show programm version information (--version)

Definition at line 58 of file console\_out.cpp.

References Program\_name, PROGVERSION, and REVISION.

Referenced by process\_cmdLine().

Here is the caller graph for this function:

#### void show\_error (const char \* *msg*, const int *errcode*, const char *help*)

Show cmdLine parser error.

Definition at line 72 of file console\_out.cpp.

References program\_name.

Referenced by internal\_error(), and process\_cmdLine().

Here is the caller graph for this function:

#### void internal\_error (const char \* *msg*)

Show cmdLine internal error.

Definition at line 86 of file console\_out.cpp.

References show\_error().

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

#### const char\* optname (const int *code*, const [ap\_Option](#AAAAAAAACQ) *options*[])

Convert code with option from cmdLine argument to char buffer.

Definition at line 97 of file console\_out.cpp.

References ap\_Option::code, and ap\_Option::name.

### Variable Documentation

#### char [PROGVERSION](#AAAAAAAAFL)[255] = "0.1"

version of programm

Definition at line 27 of file console\_out.cpp.

Referenced by show\_version().

#### char [Program\_name](#AAAAAAAAFM)[255] = "Ð¡ÐµÑ€Ð²Ð¸Ñ ÑÐ»ÑƒÐ¶ÐµÐ±Ð½Ð¾Ð¹ ÑÐ¸ÑÑ‚ÐµÐ¼Ñ‹"

name of programm

Definition at line 28 of file console\_out.cpp.

Referenced by show\_help(), and show\_version().

#### char [program\_name](#AAAAAAAAFN)[255] = "ss\_Service"

filename of programm

Definition at line 29 of file console\_out.cpp.

Referenced by show\_error(), and show\_help().

#### char [program\_year](#AAAAAAAAFO)[255] = "2010"

copyright year

Definition at line 30 of file console\_out.cpp.

## src/console\_out.h File Reference

aided functions interface header

### Functions

void [show\_help](#AAAAAAAAFQ) (const char verbose)

*Show cmdline help information (--help)*

void [show\_version](#AAAAAAAAFR) ()

*Show programm version information (--version)*

void [show\_error](#AAAAAAAAFS) (const char \*msg, const int errcode, const char help)

*Show cmdLine parser error.*

void [internal\_error](#AAAAAAAAFT) (const char \*msg)

*Show cmdLine internal error.*

const char \* [optname](#AAAAAAAAFU) (const int code, const [ap\_Option](#AAAAAAAACQ) options[])

*Convert code with option from cmdLine argument to char buffer.*

### Variables

char [PROGVERSION](#AAAAAAAAFL) []

*version of programm*

char [Program\_name](#AAAAAAAAFM) []

*name of programm*

char [program\_name](#AAAAAAAAFN) []

*filename of programm*

char [program\_year](#AAAAAAAAFO) []

*copyright year*

### Detailed Description

aided functions interface header

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [console\_out.h](#AAAAAAAAFV).

### Function Documentation

#### void show\_help (const char *verbose*)

Show cmdline help information (--help)

Definition at line 36 of file console\_out.cpp.

References program\_name, and Program\_name.

Referenced by process\_cmdLine().

Here is the caller graph for this function:

#### void show\_version ()

Show programm version information (--version)

Definition at line 58 of file console\_out.cpp.

References Program\_name, PROGVERSION, and REVISION.

Referenced by process\_cmdLine().

Here is the caller graph for this function:

#### void show\_error (const char \* *msg*, const int *errcode*, const char *help*)

Show cmdLine parser error.

Definition at line 72 of file console\_out.cpp.

References program\_name.

Referenced by internal\_error(), and process\_cmdLine().

Here is the caller graph for this function:

#### void internal\_error (const char \* *msg*)

Show cmdLine internal error.

Definition at line 86 of file console\_out.cpp.

References show\_error().

Referenced by process\_cmdLine().

Here is the call graph for this function:

Here is the caller graph for this function:

#### const char\* optname (const int *code*, const [ap\_Option](#AAAAAAAACQ) *options*[])

Convert code with option from cmdLine argument to char buffer.

Definition at line 97 of file console\_out.cpp.

References ap\_Option::code, and ap\_Option::name.

### Variable Documentation

#### char [PROGVERSION](#AAAAAAAAFL)[]

version of programm

Definition at line 27 of file console\_out.cpp.

Referenced by show\_version().

#### char [Program\_name](#AAAAAAAAFM)[]

name of programm

Definition at line 28 of file console\_out.cpp.

Referenced by show\_help(), and show\_version().

#### char [program\_name](#AAAAAAAAFN)[]

filename of programm

Definition at line 29 of file console\_out.cpp.

Referenced by show\_error(), and show\_help().

#### char [program\_year](#AAAAAAAAFO)[]

copyright year

Definition at line 30 of file console\_out.cpp.

## src/deqUdp/deqUdp.cpp File Reference

Class [deqUdp](#AAAAAAAAFX) implementation.

#include <arpa/inet.h>

#include <deque>

#include "../../../rcsLib/ortsTypes/ortsTypes.h"

#include "../buffer/ssBuffer.h"

#include "../../../udp\_port/udp\_port.h"

#include "deqUdp.h"

### Detailed Description

Class [deqUdp](#AAAAAAAAFX) implementation.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [deqUdp.cpp](#AAAAAAAAFY).

## src/deqUdp/deqUdp.h File Reference

Class [deqUdp](#AAAAAAAAFX) interface header.

### Classes

class [deqUdp](#AAAAAAAAFX)

*udp communications (based on udp\_port) with queues for listening and sending*

### Detailed Description

Class [deqUdp](#AAAAAAAAFX) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [deqUdp.h](#AAAAAAAAGA).

## src/functions/commonFuncsMgr.cpp File Reference

Class [commonFuncsMgr](#AAAAAAAAGC) interface header.

#include <pthread.h>

#include <netinet/in.h>

#include <queue>

#include "../../rcsLib/ortsTypes/ortsTypes.h"

#include "../buffer/ssBuffer.h"

#include "../../rcsLib/rcsCmd/rcsCmd.h"

#include "../../udp\_port/udp\_port.h"

#include "../srvAppLayer/functionNode/param\_desc.h"

#include "../srvAppLayer/functionNode/functionNode.h"

#include "../srvAppLayer/srvAppLayer.h"

#include "commonFuncsMgr.h"

#include "functions.h"

### Functions

void \* [equipListenPolling](#AAAAAAAAGD) (void \*user)

*Thread to polling listen udp for equipment data.*

### Detailed Description

Class [commonFuncsMgr](#AAAAAAAAGC) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [commonFuncsMgr.cpp](#AAAAAAAAGE).

### Function Documentation

#### void\* equipListenPolling (void \* *user*)

Thread to polling listen udp for equipment data.

Calls equipListenProcessing for decoding data received from equipment

##### Return values:

|  |  |
| --- | --- |
| *user* |  |

##### [Todo](#dd/da0/todo__todo000010):

Listening equipment answer - status vector:

Definition at line 35 of file commonFuncsMgr.cpp.

References app, srvAppLayer::equip\_read\_data(), srvAppLayer::equip\_reading\_event(), and srvAppLayer::terminated().

Referenced by srvAppLayer::StartListening().

Here is the call graph for this function:

Here is the caller graph for this function:

## src/functions/commonFuncsMgr.h File Reference

Class [commonFuncsMgr](#AAAAAAAAGC) interface header.

### Classes

class [commonFuncsMgr](#AAAAAAAAGC)

### *common functions manager implementation (set of function independs on target)* Functions

void \* [equipListenPolling](#AAAAAAAAGG) (void \*)

*Thread to polling listen udp for equipment data.*

### Detailed Description

Class [commonFuncsMgr](#AAAAAAAAGC) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [commonFuncsMgr.h](#AAAAAAAAGH).

### Function Documentation

#### void\* equipListenPolling (void \* *user*)

Thread to polling listen udp for equipment data.

##### [Todo](#dd/da0/todo__todo000011):

add to [commonFuncsMgr](#AAAAAAAAGC) class as static method

Calls equipListenProcessing for decoding data received from equipment

##### Return values:

|  |  |
| --- | --- |
| *user* |  |

##### [Todo](#dd/da0/todo__todo000010):

Listening equipment answer - status vector:

Definition at line 35 of file commonFuncsMgr.cpp.

References app, srvAppLayer::equip\_read\_data(), srvAppLayer::equip\_reading\_event(), and srvAppLayer::terminated().

Referenced by srvAppLayer::StartListening().

Here is the call graph for this function:

Here is the caller graph for this function:

## src/functions/specFuncsMgr.h File Reference

Class [specFuncsMgr](#AAAAAAAAGJ) interface header.

### Classes

class [specFuncsMgr](#AAAAAAAAGJ)

*special functions set manager.*

### Detailed Description

Class [specFuncsMgr](#AAAAAAAAGJ) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [specFuncsMgr.h](#AAAAAAAAGK).

## src/main.cpp File Reference

Programm entry point.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#include <pthread.h>

#include <signal.h>

#include <sys/reboot.h>

#include <deque>

#include "../../rcsLib/ortsTypes/ortsTypes.h"

#include "buffer/ssBuffer.h"

#include "../../udp\_port/udp\_port.h"

#include "arg\_parser/carg\_parser.h"

#include "console\_out.h"

#include "../../rcsLib/rcsCmd/rcsCmd.h"

#include "ICAppLayer/FunctionNode/param\_desc.h"

#include "ICAppLayer/FunctionNode/FunctionNode.h"

#include "ICAppLayer/ICAppLayer.h"

#include "Functions/CommonFuncs.h"

#include "Functions/SpecFuncs.h"

#include "Functions/functions.h"

#include "SIG\_handler.h"

#include "global.h"

### Functions

errType [process\_cmdLine](#AAAAAAAAGM) (int argc, char \*argv[])

*Parsing commandline arguments.*

errType [fileRead](#AAAAAAAAGN) (char \*fname, BYTE \*\*[buffer](#AAAAAAAAEN), size\_t \*sz)

*useless function in this programm.*

void [dbg\_hex\_print](#AAAAAAAAGO) (BYTE \*[buffer](#AAAAAAAAEN), size\_t len)

*prints hex bytes from buffer with size len .*

errType [appInit](#AAAAAAAAGP) (void)

*Initialize* [*srvAppLayer*](#AAAAAAAAGQ) *subsystem.*

errType [appDeinit](#AAAAAAAAGR) (void)

*Deinitialize* [*srvAppLayer*](#AAAAAAAAGQ) *subsystem.*

int [main](#AAAAAAAAGS) (int argc, char \*argv[])

*Programm entrypoint.*

### Detailed Description

Programm entry point.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [main.cpp](#AAAAAAAAGT).

### Function Documentation

#### errType process\_cmdLine (int *argc*, char \* *argv*[])

Parsing commandline arguments.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *argc* | - count of arguments strings |
| in | *argv[]* | - array of arguments strings |

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - if execution was successful |
| *err\_not\_found* | - if no arguments found |
| *err\_result\_error* | - if parsing was unsuccessful |

##### [Todo](#dd/da0/todo__todo000012):

reorganize process to external library

1. Define arguments type: with (*ap\_yes* ) or without (*ap\_no* ) parameters

2. Initialize arguments parser [ap\_init](#AAAAAAAADE)

3. Check for parsing errors [ap\_error](#AAAAAAAADG)

4. Execute all arguments after it parsing

get code of argument [ap\_code](#AAAAAAAADI)

switch with argument code value

execute

4. Execute only arguments with parameters after it parsing

get code of argument [ap\_code](#AAAAAAAADI)

get argument parameter [ap\_argument](#AAAAAAAADJ)

switch with argument code value

execute

Definition at line 51 of file main.cpp.

References ap\_argument(), ap\_arguments(), ap\_code(), ap\_error(), ap\_init(), ap\_no, ap\_yes, eq\_ip\_addr, eq\_udp\_listen\_port, eq\_udp\_sending\_port, equipAddr, internal\_error(), show\_error(), show\_help(), show\_version(), verbose\_level, and wUdp.

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType fileRead (char \* *fname*, BYTE \*\* *buffer*, size\_t \* *sz*)

useless function in this programm.

stays from good old times

##### [Todo](#dd/da0/todo__todo000013):

reorganize function to reading xml-files for future purposes

Definition at line 147 of file main.cpp.

#### void dbg\_hex\_print (BYTE \* *buffer*, size\_t *len*)

prints hex bytes from *buffer* with size *len* .

##### [Todo](#dd/da0/todo__todo000014):

use this function in new debug print system

Definition at line 188 of file main.cpp.

#### errType appInit (void)

Initialize [srvAppLayer](#AAAAAAAAGQ) subsystem.

result copied from [srvAppLayer::StartListening](#AAAAAAAAGU)

##### Return values:

|  |  |
| --- | --- |
| *err\_result\_ok* | - execution was successful |
| *err\_sock\_error* | - problems with communications subsystem |

Starting main programm threads [srvAppLayer::StartListening()](#AAAAAAAAGU):

1. Prepare queues for sending and listening to/from clients

2. Send & Listen threads for clients communication

3. Listen thread for equipment communication

If threads started successfully - starts service specific function initialize srvInit()

Definition at line 203 of file main.cpp.

References app, srvAppLayer::StartListening(), and verbose\_level.

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

#### errType appDeinit (void)

Deinitialize [srvAppLayer](#AAAAAAAAGQ) subsystem.

##### Return values:

|  |  |
| --- | --- |
| *always* | return err\_result\_ok, why not? |

Definition at line 230 of file main.cpp.

References app, and srvAppLayer::StopListening().

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

#### int main (int *argc*, char \* *argv*[])

Programm entrypoint.

##### Return values:

|  |  |
| --- | --- |
| *EXIT\_FAILURE* |  |
| *err\_not\_init* |  |

1. Process command line arguments **argc** and **argv** [] in [process\_cmdLine](#AAAAAAAAGM)

if arguments parsing is unsuccessfull exiting from programm

2. Check arguments:

check for missing one of exact argument

check for equipment communication settings:

* sending port need to be not equal to listen port values
* sending or listen port neet to be not equal to client listen port

check for sending port number or listening port number was far from client port number on one port number that reserved for client sending port.

3. Install system signals handlers [installSIGhandlers()](#AAAAAAAAGV)

4. Initialize application [appInit()](#AAAAAAAAGP)

4. Start functions generate from declarations

for common functions [commonFuncsMgr::startCommonFuncs()](#AAAAAAAAGW)

for special functions [specFuncsMgr::startSpecFuncs()](#AAAAAAAAGX)

5. Main programm loop [srvAppLayer::ProcessMessages()](#AAAAAAAAGY) while not terminated by signal [srvAppLayer::terminated()](#AAAAAAAAGZ)

6. Deinitialize application [appDeinit()](#AAAAAAAAGR)

Definition at line 244 of file main.cpp.

References app, appDeinit(), appInit(), AppTerminated, eq\_udp\_listen\_port, eq\_udp\_sending\_port, equipAddr, installSIGhandlers(), process\_cmdLine(), srvAppLayer::ProcessMessages(), commonFuncsMgr::startCommonFuncs(), specFuncsMgr::startSpecFuncs(), srvAppLayer::terminated(), and wUdp.

Here is the call graph for this function:

## src/SIG\_handler.cpp File Reference

System signals handlers manager.

#include <sys/types.h>

#include <unistd.h>

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <arpa/inet.h>

#include "../../rcsLib/ortsTypes/ortsTypes.h"

#include "global.h"

### Functions

void [term\_handler](#AAAAAAAAHB) (int i)

*signal* ***TERMINATE*** *handling function*

void [installSIGhandlers](#AAAAAAAAGV) (funcVoid func)

*signals handlers installer*

### Variables

funcVoid [SIGTERM\_handler](#AAAAAAAAHC)

*pointer to handling function for signal* ***TERMINATE***

### Detailed Description

System signals handlers manager.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [SIG\_handler.cpp](#AAAAAAAAHD).

### Function Documentation

#### void term\_handler (int *i*)

signal **TERMINATE** handling function

Definition at line 29 of file SIG\_handler.cpp.

References SIGTERM\_handler, and verbose\_level.

Referenced by installSIGhandlers().

Here is the caller graph for this function:

#### void installSIGhandlers (funcVoid *func*)

signals handlers installer

Definition at line 39 of file SIG\_handler.cpp.

References SIGTERM\_handler, and term\_handler().

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

### Variable Documentation

#### funcVoid [SIGTERM\_handler](#AAAAAAAAHC)

pointer to handling function for signal **TERMINATE**

Definition at line 24 of file SIG\_handler.cpp.

Referenced by installSIGhandlers(), and term\_handler().

## src/SIG\_handler.h File Reference

System signals handlers manager interface header.

### Functions

void [installSIGhandlers](#AAAAAAAAHF) (funcVoid func)

*signals handlers installer*

### Detailed Description

System signals handlers manager interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [SIG\_handler.h](#AAAAAAAAHG).

### Function Documentation

#### void installSIGhandlers (funcVoid *func*)

signals handlers installer

Definition at line 39 of file SIG\_handler.cpp.

References SIGTERM\_handler, and term\_handler().

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

## src/srvAppLayer/functionNode/functionNode.cpp File Reference

Class [functionNode](#AAAAAAAAHI) implementation.

#include <stdio.h>

#include <string.h>

#include "../../../../rcsLib/ortsTypes/ortsTypes.h"

#include "param\_desc.h"

#include "../../../../rcsLib/rcsCmd/rcsCmd.h"

#include "functionNode.h"

### Detailed Description

Class [functionNode](#AAAAAAAAHI) implementation.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [functionNode.cpp](#AAAAAAAAHJ).

## src/srvAppLayer/functionNode/functionNode.h File Reference

Class [functionNode](#AAAAAAAAHI) interface header.

### Classes

class [functionNode](#AAAAAAAAHI)

*function node interface header*

### Detailed Description

Class [functionNode](#AAAAAAAAHI) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [functionNode.h](#AAAAAAAAHL).

## src/srvAppLayer/functionNode/param\_desc.cpp File Reference

Class [param\_desc](#AAAAAAAAHN) implementation.

#include <stdio.h>

#include <string.h>

#include "../../../../rcsLib/ortsTypes/ortsTypes.h"

#include "param\_desc.h"

### Detailed Description

Class [param\_desc](#AAAAAAAAHN) implementation.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [param\_desc.cpp](#AAAAAAAAHO).

## src/srvAppLayer/functionNode/param\_desc.h File Reference

Class [param\_desc](#AAAAAAAAHN) interface header.

### Classes

class [param\_desc](#AAAAAAAAHN)

*parameter description*

### Detailed Description

Class [param\_desc](#AAAAAAAAHN) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [param\_desc.h](#AAAAAAAAHQ).

## src/srvAppLayer/srvAppLayer.cpp File Reference

Class [srvAppLayer](#AAAAAAAAGQ) implementation.

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <pthread.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <iostream>

#include <queue>

#include "../../rcsLib/ortsTypes/ortsTypes.h"

#include "../../rcsLib/rcsCmd/rcsCmd.h"

#include "../global.h"

#include "../buffer/ssBuffer.h"

#include "../../udp\_port/udp\_port.h"

#include "../deqUdp/deqUdp.h"

#include "functionNode/param\_desc.h"

#include "functionNode/functionNode.h"

#include "srvAppLayer.h"

#include "../functions/commonFuncsMgr.h"

### Functions

void \* [udpSenderThread](#AAAAAAAAHS) (void \*user)

*Thread to sending data to clients from functions answer queue.*

void \* [udpListenerThread](#AAAAAAAAHT) (void \*user)

*Thread to listening requests from clients and form queue of clients requests.*

### Variables

bool [rcvComplete\_flag](#AAAAAAAAHU) = false

*todo msc diagramm*

bool [sndAllow\_flag](#AAAAAAAAHV) = false

*todo msc diagramm*

[srvAppLayer](#AAAAAAAAGQ) \* [app](#AAAAAAAAHW)

*One global instance per application.*

### Detailed Description

Class [srvAppLayer](#AAAAAAAAGQ) implementation.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [srvAppLayer.cpp](#AAAAAAAAHX).

### Function Documentation

#### void\* udpSenderThread (void \* *user*)

Thread to sending data to clients from functions answer queue.

Thread also includes work with port opening, initializing and closing.

Definition at line 50 of file srvAppLayer.cpp.

References srvAppLayer::functionsAnswersQueue, deqUdp::sendData(), sndAllow\_flag, srvAppLayer::terminate(), srvAppLayer::terminated(), and wUdp.

Referenced by srvAppLayer::StartListening().

Here is the call graph for this function:

Here is the caller graph for this function:

#### void\* udpListenerThread (void \* *user*)

Thread to listening requests from clients and form queue of clients requests.

Thread also includes work with port opening, initializing and closing.

Definition at line 102 of file srvAppLayer.cpp.

References srvAppLayer::clientsRequestsQueue, srvAppLayer::getListenerPortNum(), rcvComplete\_flag, deqUdp::readData(), srvAppLayer::terminate(), srvAppLayer::terminated(), and wUdp.

Referenced by srvAppLayer::StartListening().

Here is the call graph for this function:

Here is the caller graph for this function:

### Variable Documentation

#### bool [rcvComplete\_flag](#AAAAAAAAHU) = false

todo msc diagramm

Flag purpose: synchronize state between receiving clients requests thread and reading for received data

Definition at line 39 of file srvAppLayer.cpp.

Referenced by srvAppLayer::ProcessMessages(), and udpListenerThread().

#### bool [sndAllow\_flag](#AAAAAAAAHV) = false

todo msc diagramm

Flag purpose: synchronize state between sending clients answers thread and preparing sending data

Definition at line 41 of file srvAppLayer.cpp.

Referenced by srvAppLayer::ProcessMessages(), and udpSenderThread().

#### [srvAppLayer](#AAAAAAAAGQ)\* [app](#AAAAAAAAHY)

One global instance per application.

Definition at line 43 of file srvAppLayer.cpp.

Referenced by appDeinit(), appInit(), equipListenPolling(), and main().

## src/srvAppLayer/srvAppLayer.h File Reference

Class [srvAppLayer](#AAAAAAAAGQ) interface header.

#include <pthread.h>

### Classes

struct [serviceState](#AAAAAAAAIA)

[*stateVector\_type*](#AAAAAAAAIB) *structural field.* struct [stateVector\_type](#AAAAAAAAIB)

*Main vector of service base states.* class [srvAppLayer](#AAAAAAAAGQ)

### *Application core layer implementaion.* Typedefs

typedef struct [serviceState](#AAAAAAAAIA) [serviceState](#AAAAAAAAIC)

typedef struct [stateVector\_type](#AAAAAAAAIB) [stateVector\_type](#AAAAAAAAID)

### Variables

[srvAppLayer](#AAAAAAAAGQ) \* [app](#AAAAAAAAHY)

*One global instance per application.*

### Detailed Description

Class [srvAppLayer](#AAAAAAAAGQ) interface header.

##### Author:

Vladimir A. Nosenko ([nosenko@ieee.org](#mailto:nosenko@ieee.org))

##### Date:

December, 2010

Copyright (c) 2010 Vladimir A.Nosenko.

The license and distribution terms for this file may be found in the file LICENSE in this distribution

Definition in file [srvAppLayer.h](#AAAAAAAAIE).

### Typedef Documentation

#### typedef struct [serviceState](#AAAAAAAAIA) [serviceState](#AAAAAAAAIA)

#### typedef struct [stateVector\_type](#AAAAAAAAIB) [stateVector\_type](#AAAAAAAAIB)

### Variable Documentation

#### [srvAppLayer](#AAAAAAAAGQ)\* [app](#AAAAAAAAHY)

One global instance per application.

Definition at line 43 of file srvAppLayer.cpp.

Referenced by appDeinit(), appInit(), equipListenPolling(), and main().

# Index

INDEX