RemoteServer

Generated by Doxygen 1.8.1.2

Wed Jun 5 2013 09:14:44

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 AcessExec Class Reference

Manages 2 different ways to access the Executer module.

.

```
#include <manager.h>
```

Public Member Functions

- **AcessExec** (boost::shared_ptr< AL::ALBroker > broker)

Public Attributes

- boost::shared_ptr< Executer > **exec**
- AL::ALProxy **pexec**

2.1.1 Detailed Description

Manages 2 different ways to access the Executer module.

.

1. Access via proxy – this is a functionality given by the framework –
2. Access via pointer to the module hence the instance of class Executer

See Also

Executer

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/manager.h
- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/manager.cpp

2.2 Behavelist Class Reference

This class manages a list of struct behaviour_t.

```
#include <behavelist.h>
```

Public Member Functions

- ∼Behavelist ()

  find an behavior at index nr of the list
- behave_t ∗ getwNr (const int &nr)

  get the size of the list
- int getSize ()

  find behavior with name
- behave_t ∗ getBehave (const string &name)

  get the starting state of the behavior with name
- int getfState (const string &name)

  get the ending state of the behavior with name
- int getlState (const string &name)

  add behavior to the ending of the list
- void addBehave (const int &fstate, const int &lstate, const string &name, const string &full)

  get content for debugging
- string **list** ()

Private Attributes

- behave_t ∗ first

  first element in the list
- behave_t ∗ last

  last element in the list

2.2.1 Detailed Description

This class manages a list of struct behaviour_t.

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/behavelist.h
- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/behavelist.cpp

2.3 behaviour_t Struct Reference

Specifies a behavior found on the robot. These behaviors are uploaded with Coreographe.

Behaviors will be treated as Events.

```
#include <behavelist.h>
```

Public Attributes

- string name

  name of the behavior without state transition
- string full

  actual filename of the behavior
- int fstate

  defined starting state for the transition
- int lstate

  defined ending state for the transition
- struct behaviour_t ∗ next

  next behavior in the list

2.3.1 Detailed Description

Specifies a behavior found on the robot. These behaviors are uploaded with Coreographe.

Behaviors will be treated as Events.

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/behavelist.h

2.4 Decoder Class Reference

Decoder: Parsing – Timeouts – Managing the Camera module – partly sending feedback to the App.

`#include <decoder.h>`

Public Member Functions

- void **setIp4** (const string &ip)
- void **setPort** (const string &iport)
- void **setPipe** (const int &pw)
- void setManager (AL::ALProxy ∗ppm)

  Timer thread, timing out the connection after 20 seconds.
- int decode (const char &toParse, event_params_t ∗ep)

  decode a command, done only in 1 character steps
- int manage (event_params_t ∗ep, boost::shared_ptr< NetNao > net, int &bat_count)

  After successfully decoding a comand it will be added to the Manager Eventlist via Manager::addCom or already executed, if easy enough.

Static Public Member Functions

- static void ∗ **timer** (void ∗args)

Private Member Functions

- bool fetch (const char &toParse, int &pos, event_params_t &ep)

  First step of parsing: Looks for valid comand specifier.
- bool getParams (const char &toParse, int &pos, event_params_t &ep, int &paramCount)

  Extracts the expected parameters to a given comand.
- void writePipe (const int &writer, const char ∗buf, const int &len)

  Write content of buf to writer.
- void sendBatteryStatus (boost::shared_ptr< NetNao > net)

  send remaining battery power to Android App
- void sendBehaviours (boost::shared_ptr< NetNao > net)

  send installed behaviors to the Android App

Private Attributes

- int pipeWrite

  writing end of the pipe to the Camera process
- string ip4

  ip of nao

- string port

    port Control Process
- AL::ALProxy ∗ **pproxyManager**

2.4.1 Detailed Description

Decoder: Parsing – Timeouts – Managing the Camera module – partly sending feedback to the App.

2.4.2 Member Function Documentation

2.4.2.1 int Decoder::decode (const char & *toParse*, event_params_t ∗ *ep*)

decode a command, done only in 1 character steps

This function uses fetch() and getParams().

Parameters

toParse	next lexem to be analysed.
ep	ep.type will be != CODE_INVALID or CODE_UNKNOWN if successful

Returns

can be CODE_UNKNOWN, CODE_VALID, CODE_INVALID

2.4.2.2 bool Decoder::fetch (const char & *toParse*, int & *pos*, event_params_t & *ep*) [private]

First step of parsing: Looks for valid comand specifier.

Parameters

toParse	next lexem to be analysed.
pos	current position in the parsed string, keep it updated.
ep	is filled with the identified comand specifier TOKEN. The TOKENs are defined in gen.h

Returns

true, when params are expected
false, otherwise

2.4.2.3 bool Decoder::getParams (const char & *toParse*, int & *pos*, event_params_t & *ep*, int & *paramCount*) [private]

Extracts the expected parameters to a given comand.

Parameters

toParse	next lexem to be analysed.
pos	current position in the parsed string, keep it updated.
ep	is filled with the corresponding param values, if valid. ep will change the code specifier TOKEN to CODE_INVALID if the parameter is invalid.
paramCount	Current count of scanned parameters

Returns

true, when more parameters or more input for the current parameter is expected.
false, when no further input is expected.

2.4.2.4 void Decoder::sendBatteryStatus (boost::shared_ptr< NetNao > *net*) `[private]`

send remaining battery power to Android App

This function uses a proxy to NetNao in broker NET_BROKER

2.4.2.5 void Decoder::sendBehaviours (boost::shared_ptr< NetNao > *net*) `[private]`

send installed behaviors to the Android App

This function uses a proxy to NetNao in broker NET_BROKER

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/decoder.h
- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/decoder.cpp

2.5 Event Class Reference

Contains a fully parsed and valid comand.

`#include <eventlist.h>`

Public Member Functions

- Event ()

  Constructor.
- Event (const Event &event)

  Copy Constructor.

Public Attributes

- const void ∗const id

  Unique identifier.
- int classification

  Current processing state of this event. Defined in gen.h.
- int taskID

  reserved
- event_params_t ep

  struct with key datas

Private Attributes

- Event ∗ next

  pointer to the next Event

Friends

- class **EventList**

2.5.1 Detailed Description

Contains a fully parsed and valid comand.

It realizes a list, the access on this list is done by EventList class.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 Event::Event ( ) `[inline]`

Constructor.

Creates an initially invalid event.

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/eventlist.h

2.6 event_params_t Struct Reference

structure which concentrates the information needed to create a valid event

`#include <eventlist.h>`

Public Attributes

- int type

  Comand Specifier TOKEN –> gen.h.
- int iparams [IPARAM_LEN]

  holds integer parameters
- float fparam

  holds float parameter
- string sparam

  holds string parameter

2.6.1 Detailed Description

structure which concentrates the information needed to create a valid event

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/eventlist.h

2.7 EventList Class Reference

Manges Access on a list of events.

`#include <eventlist.h>`

Public Member Functions

- EventList ()

  Constructor: Creates empty event list.
- ∼EventList ()

  Destructer: Deletes all events.
- bool isEmpty ()

  Check if the list is empty.
- bool hasPending ()

  Check if the list has none processed events to be executed.
- void setOrder (int ord)

  Sets priorities (order) to retrieve events, defined in gen.h.
- void addEvent (event_params_t ep)

  Adds event to the ending of the list.
- void addFirst (event_params_t ep)

  Adds event to the beginning of the list.
- void removeEvent (const void ∗const iid)

  Remove the Event with Event::id == iid.
- void removeDone ()

  Remove all finisched events.
- void removePending ()

  Remove all pending events.
- void removeAll ()

  Remove all events.
- void list ()

  Debug: list events on stdout.
- void setClassf (const void ∗const iid, int classf)

  Change the classification of the event with iid.
- int getClassf (const void ∗const iid)

  Get the classification of the event with iid.
- void setTask (const void ∗const iid, const int &tid)

  reserved
- Event ∗ getFirst ()

  Get the first event.
- Event ∗ getLast ()

  Get the last event.
- Event ∗ getPending (const bool &restart)

  Get the first Event of the list fulfilling the conditions of order, hence a given priority.
- Event withID (const void ∗const iid)

  Get the event with coresponding id.
- bool reduceLastWalking ()

  Identify a sequence of walking events and remove all walking events excluding the first and last.

Private Attributes

- boost::shared_ptr< AL::ALMutex > mutex

  Grants atomic access.
- int order

  specifies the priority to execute the events, hence retreiving them via hasPending()
- Event ∗ first

  Pointer to the first Event in the list.

- Event ∗ last

  Pointer to the last Event in the list.
- Event ∗ inspect

  Pointer to the next Event to be searched, when nothing essential hast been changed.

2.7.1 Detailed Description

Manges Access on a list of events.

Each access needs to be atomic for each instance of EventList, since it is supposed to be used in multiple threads. Modules Manger and Executer use the same instance of EventList.

2.7.2 Member Function Documentation

2.7.2.1 void EventList::addEvent (event_params_t *ep*)

Adds event to the ending of the list.

Parameters

ep	Contains key data.

2.7.2.2 void EventList::addFirst (event_params_t *ep*)

Adds event to the beginning of the list.

Parameters

ep	Contains key data.

2.7.2.3 int EventList::getClassf (const void ∗const *iid*)

Get the classification of the event with iid.

Parameters

iid	

Returns

Classification of the event.

See Also

gen.h

2.7.2.4 bool EventList::reduceLastWalking ( )

Identify a sequence of walking events and remove all walking events excluding the first and last.

Returns

true, when at least 2 walking events are existing
false, otherwise

2.7.2.5 void EventList::setClassf (const void ∗const *iid*, int *classf*)

Change the classification of the event with iid.

Parameters

iid	
classf	

See Also

gen.h

2.7.2.6 void EventList::setOrder (int *ord*)

Sets priorities (order) to retrieve events, defined in gen.h.

Parameters

ord	

2.7.2.7 Event EventList::withID (const void ∗const *iid*)

Get the event with coresponding id.

Parameters

iid	

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/eventlist.h
- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/eventlist.cpp

2.8 exec_arg Struct Reference

Structure, which holds arguments for one execution thread created by Manager::runExecuter().

The created thread exectues the given Event.

```
#include <executer.h>
```

Public Attributes

- pthread_t id

  ID of the thread.
- int tnum

  thread number, identifies allocated space in an array for this structure.
- Event ∗ event

  Event to be executed.
- boost::shared_ptr< EventList > eventList

  Access to the Eventlist, functions are atomic.
- boost::shared_ptr< AL::ALMutex > mutex

  mutex to create mutual executions between more than one of those threads.

2.8.1 Detailed Description

Structure, which holds arguments for one execution thread created by Manager::runExecuter().

The created thread exectues the given Event.

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/executer.h

2.9 Executer Class Reference

Resolves Conflicts between Events and executes them.

`#include <executer.h>`

Public Member Functions

- void setStateMan (int ∗abs, int ∗itrans, int ∗gblock, bool ∗b, bool ∗pb)

  set up pointers to stateAbs inTransition blockGen block parblock
- Executer (boost::shared_ptr< AL::ALBroker > broker, const string &name)

  Constructor.
- virtual void init ()

  Will be called right after the constructor.
- void initEventList (boost::shared_ptr< EventList > eL)

  intializes eventList with eL
- void initBehavelist (boost::shared_ptr< Behavelist > bL)

  intializes blist with bL
- int processConflicts (const Event &event)

  Resolves event - conflicts and executes one fitting event.
  Sets the #state of the roboter after the event has been executed.
- void **executerRespond** ()
- void initWalk ()

  inits position to be "ready to walk".
- void initSecure ()

  will set the roboter to a resting position with motors turned of.
- void killBehaviors ()

  kills all running behaviours
- void killRemainingTasks ()

  kill all remaining tasks
- void behave_stand ()

  move roboter to standing position
- void behave_sit ()

  move roboter to sitting position
- void behave_hello ()

  lets the roboter wave and say hello
- void behave_dance ()

  lets the roboter dance
- void behave_wipe ()

  roboter wipes out the world, after that he wipes his forehead
- void behave_gen (const string &com)

  executes a generic behavior with simplified name com

- void walk (const Event &event)

  The roboter will walk.
- void cbPoseChanged (const string &eventName, const string &postureName, const string &subscriber-Identifier)

  Callback to retrieve the physical state of the robot.
- void speak (const string &msg)

  roboter speaks string given in msg
- void moveHead (const Event &event)

  robot will move his head
- void moveArm (const Event &event)

  robot will move his arm
- void sendBatteryStatus ()

  send remaining battery power to Android App
- void sendState ()

  send roboter state to Android App

Static Public Member Functions

- static void mark_thread_done (struct exec_arg ∗aargs)

  sets exec_arg::event = 0, therefor marking the thread as done and ready for freeing allocated space
- static void ∗ execute (void ∗args)

  This function is used for creating an Execution Thread in Manager::runExecuter.

Public Attributes

- string cbip

  ip of the Control Broker
- unsigned short cbport

  port of the Control Broker
- bool cbcall

  identifies the execution of a Executer::cbPoseChanged() callback
- bool cbinc

  identifies the first detection of STATE_UNKNOWN, and therefor an unsuspected behavior
- state_t cbstate

  physical state of the robot, set through sensors, hence a callback –> cbPoseChanged()

Static Public Attributes

- static Executer ∗ self = 0

  needed in static method Executer::execute(), which is created as Thread

Private Member Functions

- int unblockfor (const int &code)

  unblocks parallel and absolute Events

Private Attributes

- boost::shared_ptr< AL::ALMutex > mutex

  mutex to grant consistent data –> espec. used in Executer::process()
- boost::shared_ptr< AL::ALMutex > sync

  deprecated
- boost::shared_ptr< EventList > eventList

  contains events to be executed
- boost::shared_ptr< Behavelist > blist

  List holding all beheaviours.
- string mpose

  physical state of the robot as string, also retrieved through the callback –> cbPoseChanged()
- AL::ALMemoryProxy mem

  Proxy to the atomic memory management of the framework.
- int ∗ stateAbs

  Pointer to logical state of the robot of class Manager.
- int ∗ inTransition

  Pointer to transition identifier of class Manager.
- int ∗ blockGen

  deprecated
- bool ∗ block

  pointer to array of blocked absolute (atomic) Events of class Manager
- bool ∗ parblock

  pointer to array of blocked parallel Events of class Manager

2.9.1 Detailed Description

Resolves Conflicts between Events and executes them.

The Executer module will be autoloaded during boot-time as part of the central naoqi-broker. It contains functions used by the Manager module to execute scheduled Events. It provides parallelism of scheduled Events.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 Executer::Executer (boost::shared_ptr< AL::ALBroker > *broker*, const string & *name*)

Constructor.

Calls Constructor of AL::ALModule and registers all functions which should be propagated to all local modules in the same broker and all external modules connected to the broker of thiss module. Calls the constructor of the mutex's

Parameters

broker	specifies the broker this module belongs to
name	visible name of this module, via this name a proxy to this module can be opened

2.9.3 Member Function Documentation

2.9.3.1 void ∗ Executer::execute (void ∗ *args*) `[static]`

This function is used for creating an Execution Thread in Manager::runExecuter.

Parameters

args	argument as pointer of struct exec_arg

2.9.3.2 void Executer::initBehavelist (boost::shared_ptr< **Behavelist** > *bL*)

intializes blist with bL

Parameters

bL	

2.9.3.3 void Executer::initEventList (boost::shared_ptr< **EventList** > *eL*)

intializes eventList with eL

Parameters

eL	

2.9.3.4 void Executer::mark_thread_done (struct **exec_arg** ∗ *aargs*) [static]

sets exec_arg::event = 0, therefor marking the thread as done and ready for freeing allocated space

This function is used in execute()

Parameters

aargs	equal to the args parameter of method execute

2.9.3.5 void Executer::moveArm (const **Event** & *event*)

robot will move his arm

Parameters

event	holds the moving mode.

2.9.3.6 void Executer::moveHead (const **Event** & *event*)

robot will move his head

Parameters

event	holds the moving mode.

2.9.3.7 int Executer::processConflicts (const **Event** & *event*)

Resolves event - conflicts and executes one fitting event.

Sets the #state of the roboter after the event has been executed.

It is called as thread per event by Manager::runExecuter() Finds and manages conflicting events and/or states in the eventList and adapts/resolves the current event.

This function is called by process()

Parameters

event	Current event to be executed.

2.9.3.8 void Executer::sendBatteryStatus ( )

send remaining battery power to Android App

This function uses a proxy to NetNao in broker AppToNAO_BROKER

2.9.3.9 void Executer::sendState ( )

send roboter state to Android App

This function uses a proxy to NetNao in broker AppToNAO_BROKER

2.9.3.10 void Executer::setStateMan (int ∗ *abs*, int ∗ *itrans*, int ∗ *gblock*, bool ∗ *b*, bool ∗ *pb*)

set up pointers to stateAbs inTransition blockGen block parblock

This is done in Manager::runExecuter()

2.9.3.11 void Executer::speak (const string & *msg*)

roboter speaks string given in msg

Parameters

msg	

2.9.3.12 int Executer::unblockfor (const int & *code*) `[private]`

unblocks parallel and absolute Events

Parameters

code	code of the currently finished Event must be one of the enum codes

See Also

gen.h

Returns

deprecated, useless

2.9.3.13 void Executer::walk (const Event & *event*)

The roboter will walk.

Parameters

event	holds the walking mode

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/executer.h
- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/executer.cpp

2.10 Manager Class Reference

Manages incomming comands and starts a thread to run the comand, if valid.

`#include <manager.h>`

Public Member Functions

- Manager (boost::shared_ptr< AL::ALBroker > broker, const string &name)

  Constructor.
- virtual ~Manager ()

  Destructor.
- virtual void init ()

  Will be called right after the constructor.
- void localRespond ()

  Kind of a ping to this module, to test if still living.
- void setCB (const string &bip, const int &bport)

  sets cbip and cbport
- void addCom (const int &type, const int &ip1, const int &ip2, const float &fp, const string &sp, const int &prio)

  decodes a string, it only processes one char at a time before returning
- void **initPipe** (const int &writer)
- void **initIp4** (const string &ip)
- void runExecuter ()

  Initialises the starting position of the robot and schedules valid events afterwards.

Public Attributes

- boost::weak_ptr< Manager > **managerSingleton**

Private Member Functions

- void initAbsTransition ()

  First step of parsing: Looks for valid comand specifier in given string.
- void initGenAllowed ()

  Initializes genAllowed.
- void initblist ()

  Initializes blist with all current none standart behaviors found on the robotPose also analyses them for additional transition information.
- int retrieveTrans (const int &from, const int &to, event_params_t *ep)

  fills ep for the specified transition from -> to
- int isGenAllowed (const int &gen, const int &abs)

  returns return genAllowed[gen][abs]

- int processConflicts (Event ∗event)

  Analyses scheduled Event for execution at the current state (stateAbs)
- int resolveConflict (Event ∗event, const int &from, const int &to)

  Resolves a non valid transition from -> to and creates a sequence of new transitions, which finaly result in to these are added at the beginning of the EventList and will be executed immediatly.
- int adaptEventList (const int &confresult, Event ∗event)

  Uses the return value of processConflicts()
- int blockfor (const int &code)

  blocks parallel and absolute Events
- bool isblocked (const int &code)

  check if the event is currently blocked
- bool isblocked (Event ∗event)

  Translation of isblocked(const int& code) for Comand Specfier TOKENs in gen.h.
- int new_thread (struct exec_arg ∗targ[MAX_THREADS], Event ∗event, boost::shared_ptr< EventList > eL)

  Thread creation and allocating management.
- int delete_thread (struct exec_arg ∗targ[MAX_THREADS], int t)

  Thread creation and allocating management.
- int free_done_threads (struct exec_arg ∗targ[MAX_THREADS])

  Thread creation and allocating management.
- int catch_dangling_threads (struct exec_arg ∗targ[MAX_THREADS])

  Thread creation and allocating management.

Private Attributes

- AL::ALMemoryProxy mem

  Proxy to atomic managed memory of the framework.
- AL::ALValue **lastOp**
- boost::shared_ptr< AL::ALMutex > mutex

  reserved.
- AcessExec accessExec
- boost::shared_ptr< EventList > eventList

  Pointer to instance of EventList, queueing valid comands.
- boost::shared_ptr< Behavelist > blist

  List holding all beheaviours.
- int pipeWrite

  file descriptor piping to the cam module
- int threadcount

  currently running threads
- string ip4

  ip of nao
- string cbip

  ip of the control module (localhost default)
- unsigned short cbport

  port of the control module (used for interprocess communication of the naoqi framework)
- int stateAbs

  absolute logical state of the nao
- int inTransition

  identifies if nao is in Transition
- int blockGen

  deprecated
- bool block [NUM_CODES]

array of blocked absolute (atomic) Events of class Manager

- bool parblock [NUM_CODES]

  array of blocked parallel Events of class Manager

- event_params_t absTransition [NUM_ABS_STATES][NUM_ABS_STATES]

  holds state transitions, for identifieing valid transitions and resolving invalid ones

- int genAllowed [NUM_GEN_STATES][NUM_ABS_STATES]

  holds allowed parallel Events to a running transition

2.10.1 Detailed Description

Manages incomming comands and starts a thread to run the comand, if valid.

The Manager module will be autoloaded during boot-time as part of the central naoqi-broker. It receives the Input from the external [AppToNAO_BROKER] in main.cpp via the NetNao module. It hands valid comands over to the Executer module.

2.10.2 Constructor & Destructor Documentation

2.10.2.1 Manager::Manager (boost::shared_ptr$<$ AL::ALBroker $>$ *broker,* const string & *name*)

Constructor.

Calls Constructor of AL::ALModule and registers all functions which should be propagated to all local modules in the same broker and all external modules connected to the broker of this module. Initializes eventList for itself and Executer module

Parameters

broker	specifies the broker this module belongs to
name	visible name of this module, via this name a proxy to this module can be opened

2.10.3 Member Function Documentation

2.10.3.1 int Manager::adaptEventList (const int & *confresult,* Event $*$ *event*) `[private]`

Uses the return value of processConflicts()

Uses the return value of processConflicts()

0, use resolveConflict() 1, do nothing -1, do nothing -2, do nothing -3, discard parallel event

2.10.3.2 void Manager::addCom (const int & *type,* const int & *ip1,* const int & *ip2,* const float & *fp,* const string & *sp,* const int & *prio*)

decodes a string, it only processes one char at a time before returning

The current decoding stage is managed via states, the stages are defined in gen.h STG_FETCH -> STG_PARAM -> STG_VALID It is called by the [AppToNAO_BROKER] via a proxy to the Manager module belonging to naoqi broker.

Parameters

toParse	String to be scanned and translated.

Returns

Returns current scanning position or -1, when the Connection is to be closed. adds an Event to Eventlist

prio = 0 -> add to the beginning

prio = 1 -> add to the ending

parameters correspond to the members of event_params_t

2.10.3.3 int Manager::blockfor (const int & *code*) [private]

blocks parallel and absolute Events

Parameters

code	code of the Event to be started must be one of the enum codes

See Also

gen.h

Returns

deprecated, useless

2.10.3.4 void Manager::initAbsTransition ( ) [private]

First step of parsing: Looks for valid comand specifier in given string.

Parameters

toParse	String to be scanned.
pos	initial starting position and ending position after returning.
ep	is filled with the identified comand specifier TOKEN. The TOKENs are defined in gen.h

Returns

true, when params are expected
false, otherwise Extracts the expected parameters to a given comand.

Parameters

toParse	String to be scanned.
pos	Position – Index – to start scanning
ep	is filled with the corresponding param values, if valid. ep will change the code specifier TOKEN to CODE_INVALID if the parameter is invalid.
paramCount	Current count of scanned parameters

Returns

true, when more parameters or more input for the current parameter is expected.
false, when no further input is expected. Initializes absTransition

2.10.3.5 bool Manager::isblocked (const int & *code*) [private]

check if the event is currently blocked

Parameters

code	must be one of the enum codes

See Also

gen.h

2.10.3.6 int Manager::isGenAllowed (const int & *gen,* const int & *abs*) [private]

returns return genAllowed[gen][abs]

Parameters

gen	one of the specified parallel Events specified in enum state_gen in gen.h
abs	current state, specified in enum state_abs in gen.h

Returns

-1, from or to is out of bound
0, is not allowed during that state
1, is allowd during that state

2.10.3.7 int Manager::processConflicts (Event ∗ *event*) [private]

Analyses scheduled Event for execution at the current state (stateAbs)

Returns

0, absolute event has to be resolved to be executed 1, no conflict
-1, absolute event currently blocked
-2, parallel event currently blocked
-3, parallel event is not allowed in current state

2.10.3.8 int Manager::resolveConflict (Event ∗ *event,* const int & *from,* const int & *to*) [private]

Resolves a non valid transition from -> to and creates a sequence of new transitions, which finaly result in *to* these are added at the beginning of the EventList and will be executed immediatly.

Returns

-1, no resolved transitons sequence could be found
0, resolved transition sequence has been added to the EventList

2.10.3.9 int Manager::retrieveTrans (const int & *from,* const int & *to,* event_params_t ∗ *ep*) [private]

fills ep for the specified transition from -> to

Parameters

from	state, defined in enum state_abs in gen.h
to	state, defined in enum state_abs in gen.h

Returns

0, transition is valid, ep holds a fully specified transition
1, transition is valid, ep does not hold an Event (Events needing additional parameters) -1, from or to is out of bound (doesn't actually exist)
-2, transition is invalid

2.10.3.10 void Manager::runExecuter ( )

Initialises the starting position of the robot and schedules valid events afterwards.

It is called by the [NET_BROKER] via a proxy to the Manager module as a thread. This thread lives until the Manager module is killed. It runs parallel to the decoding and executes events – valid comands – as new threads.

2.10.4 Member Data Documentation

2.10.4.1 AcessExec Manager::accessExec [private]

See Also

AcessExec

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/manager.h
- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/manager.cpp

2.11 NetNao Class Reference

Externel module – with own broker – managing the android and roboter connectivities.

`#include <netNao.h>`

Public Member Functions

- NetNao (boost::shared_ptr< AL::ALBroker > broker, const string &name)
  Constructor.
- virtual ∼NetNao ()
  Destructor.
- virtual void init ()
  Will be called right after the constructor.
- void **writePipe** (const int &writer, const AL::ALValue &buf, const int &len)
- int bindTcp (const string &port)
  bind the server to specified port
- void singleListen (const int &sockServer)
  Listen for incomming connections.
- int acceptClient (const int &sockServer)
  Supposed to be called after singleListen(), accepts the client. Connection is now established.

- int sendString (const int &sockClient, const AL::ALValue &buf, const int &len, const int &indexStart)

  sends a string with given length to the connected client
- int recvData (const int &sockClient, const boost::shared_ptr< char * > &buf, const unsigned int &len, const unsigned int &indexStart)

  receives data with given max length from the connected client
- void disconnect (const int &sockClient)

  Disconnects the Client.
- void **ckill** (const int &sockClient)
- void unbind (const int &sockServer)

  Closes server socket.
- int getClient_tcp ()
- int getServer_tcp ()
- int **getMode** ()

Public Attributes

- string **ip4**

Private Attributes

- int sclient_tcp

  Socket of connected client.
- int sserver_tcp

  Socket of the server.
- int **mode**

2.11.1 Detailed Description

Externel module – with own broker – managing the android and roboter connectivities.

The module is created and used during runtime of its main broker [AppToNAO_BROKER] in main.cpp

2.11.2 Constructor & Destructor Documentation

2.11.2.1 NetNao::NetNao (boost::shared_ptr< AL::ALBroker > *broker,* const string & *name*)

Constructor.

Calls Constructor of AL::ALModule and registers all functions which should be propagated to all local modules in the same broker and all external modules connected to the broker of thiss module.

Parameters

broker	specifies the broker this module belongs to
name	visible name of this module, via this name a proxy to this module can be opened

2.11.3 Member Function Documentation

2.11.3.1 int NetNao::acceptClient (const int & *sockServer*)

Supposed to be called after singleListen(), accepts the client. Connection is now established.

Parameters

sockServer	returned server socket by bindTcp() returns the client socket hence sclient_tcp

2.11.3.2 int NetNao::bindTcp (const string & *port*)

bind the server to specified port

Parameters

port	to be bound

Returns

returns the server socket hence sserver_tcp

2.11.3.3 void NetNao::disconnect (const int & *sockClient*)

Disconnects the Client.

Parameters

sockClient	client socket returned by acceptClient()

2.11.3.4 int NetNao::getClient_tcp ( )

Returns

Currently connected client socket.

2.11.3.5 int NetNao::getServer_tcp ( )

Returns

Current server socket.

2.11.3.6 int NetNao::recvData (const int & *sockClient,* const boost::shared_ptr< char * > & *buf,* const unsigned int & *len,* const unsigned int & *indexStart*)

receives data with given max length from the connected client

This call is blocking.

Parameters

sockClient	client socket returned by acceptClient()
buf	buffer
len	max length of the buffer
indexStart	specifies the index of the buffer to be written first

Returns

sucess: number charackters received
fail: -1 on invalid socket, SOCK_CLOSED or SOCK_LOST on resetted conection

2.11.3.7 int NetNao::sendString (const int & *sockClient*, const AL::ALValue & *buf*, const int & *len*, const int & *indexStart*)

sends a string with given length to the connected client

This call is blocking.

Parameters

sockClient	client socket returned by acceptClient()
buf	buffer holding the string
len	length of the string
indexStart	specifies first charackter to be sent

Returns

sucess: number charackters sent
fail: -1

2.11.3.8 void NetNao::singleListen (const int & *sockServer*)

Listen for incomming connections.

Only one connection is allowed on incomming queue. This Call is blocking.

Parameters

sockServer	returned server socket by bindTcp()

2.11.3.9 void NetNao::unbind (const int & *sockServer*)

Closes server socket.

Parameters

sockClient	client socket returned by acceptClient()

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/netNao.h
- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/netNao.cpp

2.12 thread_arg Struct Reference

Thread argument structure for the actual camera sending thread.

Public Attributes

- pthread_t **id**
- int **tnum**
- char **appIp** [IP_LEN+1]
- unsigned short **appPort**
- int **sclient**
- AL::ALValue **image**
- string **nameId**
- AL::ALVideoDeviceProxy ∗ **proxyCam**

2.12.1 Detailed Description

Thread argument structure for the actual camera sending thread.

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/main.cpp

2.13 timer_arg Struct Reference

Arguments for the timer thread Decoder::timer(void$*$ args).

```
#include <decoder.h>
```

Public Attributes

- pthread_t id

  id of the thread
- int tnum

  thread number, identifies allocated space in an array for this structure.
- boost::shared_ptr< int > bat_count

  pointer to the count of received BAT messages
- boost::shared_ptr< NetNao > net

  pointer to the NetNao module

2.13.1 Detailed Description

Arguments for the timer thread Decoder::timer(void$*$ args).

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

- /home/jierr/development/naoqi-sdk-1.12.3-linux64/projects/RemoteNAO/remoteServer/decoder.h