

Entwicklerdokumentation

Service-Interface für ein Formula-Student-Fahrzeug

**Technische Universität Ilmenau
Softwareprojekt SS 2013
Gruppe 19**

Christian Boxdörfer
Thomas Golda
Daniel Häger
David Kudlek
Tom Porzig
Tino Tausch
Tobias Zehner
Sebastian Zehnter

19.06.2013

betreut durch

Dr. Heinz-Dietrich Wuttke, TU Ilmenau
Oliver Dittrich, fachlicher Betreuer Team StarCraft e.V.

Inhaltsverzeichnis

1	Hierarchie-Verzeichnis	1
1.1	Klassenhierarchie	1
2	Klassen-Verzeichnis	2
2.1	Auflistung der Klassen	2
3	Klassen-Dokumentation	3
3.1	CommunicatingSocket Klassenreferenz	3
3.1.1	Ausführliche Beschreibung	4
3.1.2	Dokumentation der Elementfunktionen	4
3.1.2.1	connect	4
3.1.2.2	getForeignAddress	4
3.1.2.3	getForeignPort	4
3.1.2.4	recv	4
3.1.2.5	send	5
3.2	Data Klassenreferenz	5
3.2.1	Ausführliche Beschreibung	5
3.2.2	Beschreibung der Konstruktoren und Destruktoren	6
3.2.2.1	Data	6
3.2.3	Dokumentation der Elementfunktionen	6
3.2.3.1	getDatatype	6
3.2.3.2	getPosition	6
3.2.3.3	getValue	6
3.3	DatatypeDaemon Klassenreferenz	6
3.3.1	Ausführliche Beschreibung	7
3.3.2	Beschreibung der Konstruktoren und Destruktoren	7
3.3.2.1	DatatypeDaemon	7
3.3.3	Dokumentation der Elementfunktionen	7
3.3.3.1	getPosActualPacket	7
3.3.3.2	getTime	7
3.3.3.3	parseNextValue	7
3.4	dbData Klassenreferenz	8
3.4.1	Ausführliche Beschreibung	8
3.4.2	Beschreibung der Konstruktoren und Destruktoren	8
3.4.2.1	dbData	9
3.4.3	Dokumentation der Elementfunktionen	9
3.4.3.1	getDB	9
3.4.3.2	getHost	9
3.4.3.3	getPW	9

3.4.3.4	getUser	9
3.5	DBPacketInsert Klassenreferenz	9
3.5.1	Ausführliche Beschreibung	10
3.5.2	Dokumentation der Elementfunktionen	10
3.5.2.1	db_insert	10
3.6	Decoder Klassenreferenz	10
3.6.1	Beschreibung der Konstruktoren und Destruktoren	10
3.6.1.1	Decoder	11
3.6.1.2	Decoder	12
3.6.2	Dokumentation der Elementfunktionen	12
3.6.2.1	getNextData	12
3.6.2.2	getPackageNum	12
3.6.2.3	getPackagePos	12
3.7	Encoder Klassenreferenz	13
3.7.1	Ausführliche Beschreibung	13
3.7.2	Beschreibung der Konstruktoren und Destruktoren	13
3.7.2.1	Encoder	13
3.7.3	Dokumentation der Elementfunktionen	14
3.7.3.1	getNextPackage	14
3.7.3.2	getPackage	14
3.7.3.3	getPackageSize	14
3.7.3.4	getPackageSum	15
3.8	Insert Klassenreferenz	15
3.8.1	Ausführliche Beschreibung	15
3.8.2	Beschreibung der Konstruktoren und Destruktoren	15
3.8.2.1	Insert	15
3.8.3	Dokumentation der Elementfunktionen	15
3.8.3.1	insertIntoDB	15
3.9	Location Klassenreferenz	16
3.9.1	Ausführliche Beschreibung	16
3.9.2	Beschreibung der Konstruktoren und Destruktoren	16
3.9.2.1	Location	16
3.9.3	Dokumentation der Elementfunktionen	16
3.9.3.1	getAddress	16
3.9.3.2	getPort	17
3.10	Socket Klassenreferenz	17
3.10.1	Ausführliche Beschreibung	18
3.10.2	Beschreibung der Konstruktoren und Destruktoren	18
3.10.2.1	~Socket	18
3.10.3	Dokumentation der Elementfunktionen	18
3.10.3.1	getLocalAddress	18
3.10.3.2	getLocalPort	18
3.10.3.3	resolveService	18
3.10.3.4	setLocalAddressAndPort	19
3.10.3.5	setLocalPort	19
3.11	SocketException Klassenreferenz	19
3.11.1	Ausführliche Beschreibung	20
3.11.2	Beschreibung der Konstruktoren und Destruktoren	20

3.11.2.1	SocketException	20
3.11.2.2	~SocketException	20
3.11.3	Dokumentation der Elementfunktionen	20
3.11.3.1	what	20
3.12	T_nuex Strukturreferenz	20
3.13	TCPServerSocket Klassenreferenz	21
3.13.1	Ausführliche Beschreibung	21
3.13.2	Beschreibung der Konstruktoren und Destruktoren	21
3.13.2.1	TCPServerSocket	21
3.13.2.2	TCPServerSocket	22
3.13.3	Dokumentation der Elementfunktionen	22
3.13.3.1	accept	22
3.14	TCPSocket Klassenreferenz	22
3.14.1	Ausführliche Beschreibung	23
3.14.2	Beschreibung der Konstruktoren und Destruktoren	23
3.14.2.1	TCPSocket	23
3.14.2.2	TCPSocket	23
3.15	UDPSocket Klassenreferenz	24
3.15.1	Ausführliche Beschreibung	25
3.15.2	Beschreibung der Konstruktoren und Destruktoren	25
3.15.2.1	UDPSocket	25
3.15.2.2	UDPSocket	25
3.15.2.3	UDPSocket	25
3.15.3	Dokumentation der Elementfunktionen	25
3.15.3.1	disconnect	25
3.15.3.2	joinGroup	26
3.15.3.3	leaveGroup	26
3.15.3.4	recvFrom	26
3.15.3.5	sendTo	27
3.15.3.6	setMulticastTTL	27

1 Hierarchie-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Data	5
DatatypeDaemon	6
dbData	8
DBPacketInsert	9
Decoder	10
Encoder	13
exception	
SocketException	19
Insert	15
Location	16
Socket	17
CommunicatingSocket	3
TCPSocket	22
UDPSocket	24
TCPServerSocket	21
T_nuex	20

2 Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

CommunicatingSocket	3
Data	5
DatatypeDaemon	
Klasse zum Aufbereiten der Daten	6
dbData	
Klasse zum Auslesen der Zugangsdaten für die Datenbank	8
DBPacketInsert	9
Decoder	10
Encoder	13
Insert	
Klasse zum Aufbauen einer Verbindung zur Datenbank und Einfügen von Daten	15
Location	16
Socket	17
SocketException	19
T_nuex	20
TCPServerSocket	21
TCPSocket	22
UDPSocket	24

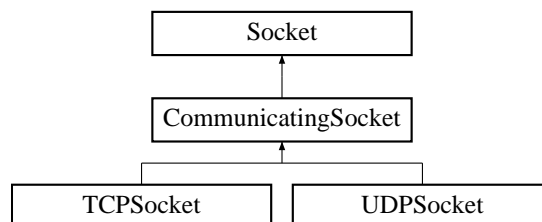
3 Klassen-Dokumentation

Wir bedienen uns teilweise an der PracticalSocket-Bibliothek um die Socket-Kommunikation unter C++ zu vereinfachen. Der Quellcode in `PracticalSocket.h` und `PracticalSocket.cpp` stammt bis auf ein paar kleinere Änderungen also nicht von uns. Der Quelltext wurde hier entnommen: <http://cs.ecs.baylor.edu/~donahoo/practical/CSockets/practical/>

3.1 CommunicatingSocket Klassenreferenz

```
#include <PracticalSocket.h>
```

Klassendiagramm für CommunicatingSocket:



Öffentliche Methoden

- void `connect` (const string &foreignAddress, unsigned short foreignPort) throw (SocketException)
- void `send` (const void *buffer, int bufferLen) throw (SocketException)
- int `recv` (void *buffer, int bufferLen) throw (SocketException)
- string `getForeignAddress` () throw (SocketException)
- unsigned short `getForeignPort` () throw (SocketException)

Geschützte Methoden

- **CommunicatingSocket** (int type, int protocol) throw (SocketException)
- **CommunicatingSocket** (int newConnSD)

Weitere Geerbte Elemente

3.1.1 Ausführliche Beschreibung

[Socket](#) der in der Lage ist sich zu verbinden, zu senden und zu empfangen.

3.1.2 Dokumentation der Elementfunktionen

3.1.2.1 void CommunicatingSocket::connect (const string & *foreignAddress*, unsigned short *foreignPort*) throw SocketException)

Baut eine Verbindung über Sockets zu einem bestimmten Teilnehmer.

Parameter

<i>foreign-Address</i>	Adresse des Teilnehmers (IP Adresse oder Name).
<i>foreignPort</i>	Portnummer des Teilnehmers.

Ausnahmebehandlung

SocketException	wird geworfen wenn die Verbindung fehlschlägt.
---------------------------------	--

3.1.2.2 string CommunicatingSocket::getForeignAddress () throw SocketException)

Holt die Zieladresse.

Rückgabe

Zieladresse.

Ausnahmebehandlung

SocketException	wird geworfen falls das Holen fehlschlägt.
---------------------------------	--

3.1.2.3 unsigned short CommunicatingSocket::getForeignPort () throw SocketException)

Holt den Zielport. Call [connect\(\)](#) before calling [recv\(\)](#)

Rückgabe

Zielportnummer.

Ausnahmebehandlung

SocketException	wird geworfen falls das Holen fehlschlägt.
---------------------------------	--

3.1.2.4 int CommunicatingSocket::recv (void * *buffer*, int *bufferLen*) throw SocketException)

Liest eine Nachricht in den übergebenen Buffer ein. [connect\(\)](#) muss vorher aufgerufen werden.

Parameter

<i>buffer</i>	Speicher in den die Nachricht geschrieben werden soll.
<i>bufferLen</i>	Maximale Anzahl an Bytes die empfangen werden sollen.

Rückgabe

Gibt die Anzahl der gelesenen Bytes zurück. 0 wenn EOF und -1 im Fehlerfall.

Ausnahmebehandlung

SocketException	wird geworfen falls keine Daten empfangen werden können.
---------------------------------	--

3.1.2.5 void CommunicatingSocket::send (const void * *buffer*, int *bufferLen*) throw **SocketException**)

Schreibt den übergebenen Buffer in den [Socket](#). Vor [send\(\)](#) muss [collect\(\)](#) aufgerufen werden.

Parameter

<i>buffer</i>	Speicher in den geschrieben werden soll.
<i>bufferLen</i>	Anzahl der Bytes die geschrieben werden sollen.

Ausnahmebehandlung

SocketException	wird geworfen wenn keine Daten gesendet werden können.
---------------------------------	--

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PracticalSocket.h
- PracticalSocket.cpp

3.2 Data Klassenreferenz

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

Öffentliche Methoden

- [Data](#) (double value, unsigned int datatype, unsigned int position)
- double [getValue](#) ()
- unsigned int [getDatatype](#) ()
- unsigned int [getPosition](#) ()

3.2.1 Ausführliche Beschreibung

Datenstruktur, die einen Fahrzeugwert und die dazugehörigen Daten speichert.

3.2.2 Beschreibung der Konstruktoren und Destruktoren

3.2.2.1 Data::Data (double *value*, unsigned int *datatype*, unsigned int *position*)

Erzeugt eine Datenstruktur zur Speicherung von Fahrzeugdaten.

Parameter

<i>value</i>	Wert des Datensatzes.
<i>datatype</i>	Datentyp des Datensatzes.
<i>position</i>	Position des Datensatzes in den ursprünglichen Daten.

3.2.3 Dokumentation der Elementfunktionen

3.2.3.1 unsigned int Data::getDatatype ()

Rückgabe

Gibt den Datentyp des Datensatzes zurück.

3.2.3.2 unsigned int Data::getPosition ()

Rückgabe

Gibt die Position des Datensatzes in den ursprünglichen Daten zurück.

3.2.3.3 double Data::getValue ()

Rückgabe

Gibt den Wert des Datensatzes zurück.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Data.h
- Data.cpp

3.3 DatatypeDaemon Klassenreferenz

Klasse zum Aufbereiten der Daten.

```
#include <DatatypeDaemon.h>
```

Öffentliche Methoden

- [DatatypeDaemon](#) ()
- string [parseNextValue](#) ()

- int [getPosActualPacket](#) ()
- int [getTime](#) ()

3.3.1 Ausführliche Beschreibung

Klasse zum Aufbereiten der Daten.

In dieser Klasse werden die Daten nicht nur in das richtige Format umgewandelt, sondern auch durch eine Zehnerpotenz geteilt, sodass sie wieder ihrem Wert vor der Übertragung entsprechen. Es werden weiterhin Zusatzinformationen angeboten, die es sowohl ermöglichen, die Paketnummer und damit die Tabelle, in die eingefügt werden soll, zu ermitteln, als auch den Zeitstempel der Daten einzufügen. Diese Klasse dient also als Schnittstelle für [DBPacketInsert](#) zum [Decoder](#).

Siehe auch

[Data](#)

3.3.2 Beschreibung der Konstruktoren und Destruktoren

3.3.2.1 DatatypeDaemon::DatatypeDaemon ()

Konstruktor der Klasse.

3.3.3 Dokumentation der Elementfunktionen

3.3.3.1 int DatatypeDaemon::getPosActualPacket ()

Diese Funktion liefert die Position des jetzigen Datenwerts im Übertragungsvektor der Mikro-AutoboxII zurück.

Dies ist für die Bestimmung des Übertragungspakets vom Embedded PC wichtig, da darauf basierend die richtige Datenbankanfrage in [DBPacketInsert](#) aufgebaut wird.

Rückgabe

Aus [Data](#) extrahierter und aufbereiteter Wert im String-Format.

3.3.3.2 int DatatypeDaemon::getTime ()

Diese Memberfunktion gibt den Zeitstempel des aktuellen Pakets zurück.

Rückgabe

Zeitstempel des Pakets.

3.3.3.3 string DatatypeDaemon::parseNextValue ()

Diese Funktion wird von [DBPacketInsert](#) aufgerufen.

Sie holt sich ein [Data](#) Objekt vom [Decoder](#) und extrahiert daraus die Nutzdaten und die Position dieser Daten im Paket. Die Position wird dabei in einer Membervariablen zwischengespeichert. Nachdem der Datentyp bestimmt wurde, wird eine Division durch eine Zehnerpotenz entsprechend dem jetzt erst extrahierten Wert aus [Data](#) durchgeführt, um den Wert auf seine ursprüngliche Größe zurückzubringen. Dies wird jedoch nur ausgeführt, wenn der Wert vor der Übertragung ein Gleitkommawert war. Abschließend wird der Wert in einen String umgewandelt, der dann an den Aufrufer zurückgegeben wird.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- DatatypeDaemon.h
- DatatypeDaemon.cpp

3.4 dbData Klassenreferenz

Klasse zum Auslesen der Zugangsdaten für die Datenbank.

```
#include <dbData.h>
```

Öffentliche Methoden

- [dbData](#) ()
- string [getHost](#) ()
Memberfunktion, die die Hostadresse bereitstellt.
- string [getUser](#) ()
Memberfunktion, die den Benutzernamen für die Datenbank bereitstellt.
- string [getPW](#) ()
Memberfunktion, die das Passwort der Datenbank bereitstellt.
- string [getDB](#) ()
Memberfunktion, die den Namen der Datenbank bereitstellt.

3.4.1 Ausführliche Beschreibung

Klasse zum Auslesen der Zugangsdaten für die Datenbank.

Diese Klasse liest die Zugangsdaten aus einer Textdatei (dbconfig.txt) aus und stellt sie über Memberfunktionen bereit.

Siehe auch

[Insert](#)

3.4.2 Beschreibung der Konstruktoren und Destruktoren

3.4.2.1 dbData::dbData ()

Konstruktor der Klasse. Hier werden die Zugangsdaten aus der Textdatei in die entsprechenden lokalen Variablen eingelesen.

3.4.3 Dokumentation der Elementfunktionen

3.4.3.1 string dbData::getDB ()

Memberfunktion, die den Namen der Datenbank bereitstellt.

Rückgabe

Der Name der Datenbank.

3.4.3.2 string dbData::getHost ()

Memberfunktion, die die Hostadresse bereitstellt.

Wird von [Insert](#) aufgerufen.

Rückgabe

Die Hostadresse.

3.4.3.3 string dbData::getPW ()

Memberfunktion, die das Passwort der Datenbank bereitstellt.

Rückgabe

Das Passwort für die Datenbank und den Nutzer.

3.4.3.4 string dbData::getUser ()

Memberfunktion, die den Benutzernamen für die Datenbank bereitstellt.

Rückgabe

Der Nutzernamen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- dbData.h
- dbData.cpp

3.5 DBPacketInsert Klassenreferenz

```
#include <DBPacketInsert.h>
```

Öffentliche Methoden

- [DBPacketInsert](#) ()
Konstruktor der Klasse.
- void [db_insert](#) ()

3.5.1 Ausführliche Beschreibung

Klasse zum Aufbau eines SQL-Strings.

Diese Klasse baut einen SQL-String mit Hilfe anderer Klassen auf, mit dem schließlich die Daten in die Datenbank geschrieben werden.

Siehe auch

[DatatypeDaemon](#)
[Insert](#)

3.5.2 Dokumentation der Elementfunktionen

3.5.2.1 void DBPacketInsert::db_insert ()

Memberfunktion zum Aufruf aus anderen Klassen.

Diese Funktion löst den Aufruf von Unterfunktionen aus, die dann den SQL-String modular aufbauen und schließlich an [Insert](#) übergeben. Gleichzeitig bietet sie die Schnittstelle zu den anderen Teilen des Programms.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- DBPacketInsert.h
- DBPacketInsert.cpp

3.6 Decoder Klassenreferenz

Öffentliche Methoden

- [Decoder](#) (char *buffer, const int bufferlen)
- [Decoder](#) (char *buffer, const int bufferlen, char *vecLayout, const int vecLayoutlen, char *vecDatatypes, const int vecDatatypeslen, char *vecComma, const int vecCommalen)
- [Data getNextData](#) ()
- unsigned int [getPackageNum](#) ()
- unsigned int [getPackagePos](#) (char *vecLayout, const int vecLayoutlen)

3.6.1 Beschreibung der Konstruktoren und Destruktoren

3.6.1.1 Decoder::Decoder (char * *buffer*, const int *bufferlen*)

Erzeugt einen Dekoder, der zum Dekodieren der Paketinformation dient.

Parameter

<i>buffer</i>	Speicher, der die Paketinformationen enthält. [Layout,Datentypen,Kommasetzung]
<i>bufferlen</i>	Länge von <i>buffer</i> .

3.6.1.2 Decoder::Decoder (char * *buffer*, const int *bufferlen*, char * *vecLayout*, const int *vecLayoutlen*, char * *vecDatatypes*, const int *vecDatatypeslen*, char * *vecComma*, const int *vecCommalen*)

Erzeugt einen Dekoder der ein Datenpaket anhand der übergebenen Informationen dekodiert.

Parameter

<i>buffer</i>	Speicher des Datenpakets.
<i>bufferlen</i>	Länge von <i>buffer</i> .
<i>vecLayout</i>	Aufteilung des ursprünglichen Datenstroms, die aus den Paketinformationen dekodiert wurden. Dient zur Ermittlung der konkreten Datensätze.
<i>vecLayoutlen</i>	Länge von <i>vecLayout</i> .
<i>vecDatatypes</i>	Beinhaltet die Informationen zu den Datentypen der jeweiligen Datensätze.
<i>vec-Datatypeslen</i>	Länge von <i>vecDatatypes</i> .
<i>vecComma</i>	Beinhaltet die Kommasetzung sämtlicher Datensätze.
<i>vec-Commalen</i>	Länge von <i>vecComma</i> .

3.6.2 Dokumentation der Elementfunktionen

3.6.2.1 Data Decoder::getNextData ()

Holt den nächsten Datensatz aus den empfangenen Daten.

Rückgabe

Gibt ein Datenobjekt [Data](#) zurück das sämtlich Informationen über den Datensatz enthält.

3.6.2.2 unsigned int Decoder::getPackageNum ()

Holt die Paketnummer des aktuell bearbeiteten Pakets.

Rückgabe

Paketnummer des aktuellen Pakets

3.6.2.3 unsigned int Decoder::getPackagePos (char * *vecLayout*, const int *vecLayoutlen*)

Holt die Position des aktuellen Pakets im ursprünglichen Datensatz.

Parameter

<i>vecLayout</i>	Aufteilung des ursprünglichen Datenstroms die aus den Paketinformationen dekodiert wurden.
<i>vecLayoutlen</i>	Länge von <i>vecLayout</i> .

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Encoding.h
- Encoding.cpp

3.7 Encoder Klassenreferenz

```
#include <Encoding.h>
```

Öffentliche Methoden

- [Encoder](#) (const char *buffer, const int bufferlen, const char *vecLayout, const int vecLayoutlen, const char *vecDatatypes, const int vecDatatypeslen)
- int [getPackage](#) (char *package, size_t len, unsigned short packageName)
- int [getNextPackage](#) (char *package, size_t len)
- int [getPackageSize](#) (unsigned short packageName)
- unsigned int [getPackageSum](#) ()

3.7.1 Ausführliche Beschreibung

Service, der aus einem kompletten Satz Fahrzeugdaten mehrere Pakete erzeugt und komprimiert. Die Komprimierung ist noch nicht implementiert.

3.7.2 Beschreibung der Konstruktoren und Destruktoren

3.7.2.1 `Encoder::Encoder (const char * buffer, const int bufferlen, const char * vecLayout, const int vecLayoutlen, const char * vecDatatypes, const int vecDatatypeslen)`

Erzeugt einen [Encoder](#).

Parameter

<i>buffer</i>	Die zu bearbeitenden Daten. Dabei muss es sich um einen Datenstrom handeln, in dem jeweils 2 Byte einen Fahrzeugwert entsprechen.
<i>bufferlen</i>	Die Länge der zu bearbeitenden Daten.

<i>vecLayout</i>	Gibt an, wie die Daten geteilt werden sollen. [Anfangsbyte Paket 1, Anfangsbyte Paket 2, ..., Anfangsbyte Paket n]
<i>vecLayoutlen</i>	Die Länge von <i>vecLayout</i> .
<i>vecDatatypes</i>	Gibt an, um welchen Datentyp es sich jeweils handelt.
<i>vecDatatypeslen</i>	Die Länge von <i>vecDatatypes</i> .

3.7.3 Dokumentation der Elementfunktionen

3.7.3.1 int Encoder::getNextPackage (char * *package*, size_t *len*)

Holt das jeweils nächste Paket. (1,2,...,n,1,2,...)

Parameter

<i>package</i>	Speicher in den das Paket geschrieben werden soll.
<i>len</i>	Länge von <i>package</i> .

Rückgabe

Die Länge des Pakets oder -1 falls *len* zu klein.

3.7.3.2 int Encoder::getPackage (char * *package*, size_t *len*, unsigned short *packageNumber*)

Holt ein Paket mit einer speziellen Paketnummer.

Parameter

<i>package</i>	Speicher in den das Paket geschrieben werden soll.
<i>len</i>	Länge von <i>package</i> .
<i>package-Number</i>	Paketnummer des gewünschten Pakets.

Rückgabe

Die Länge des Pakets oder -1 falls Paket mit *packageNumber* nicht vorhanden oder *len* zu klein.

3.7.3.3 unsigned int Encoder::getPackageSize (unsigned short *packageNumber*)

Gibt die Paketgröße eines speziellen Pakets zurück.

Parameter

<i>package-Number</i>	Paketnummer dessen Größe gesucht ist.
-----------------------	---------------------------------------

Rückgabe

Größe des Pakets oder -1 falls Paket mit *packageNumber* nicht vorhanden.

3.7.3.4 unsigned int Encoder::getPackageSum ()

Gibt die Anzahl der Pakete zurück.

Rückgabe

Anzahl der Pakete.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Encoding.h
- Encoding.cpp

3.8 Insert Klassenreferenz

Klasse zum Aufbauen einer Verbindung zur Datenbank und Einfügen von Daten.

```
#include <Insert.h>
```

Öffentliche Methoden

- [Insert](#) ()
- int [insertIntoDB](#) (string *anw)

3.8.1 Ausführliche Beschreibung

Klasse zum Aufbauen einer Verbindung zur Datenbank und Einfügen von Daten.

In dieser Klasse wird die eigentliche Kommunikation mit der MySQL-Datenbank realisiert. Hierzu wird der MySQL Connector/C++ benutzt. Die Zugangsdaten werden dafür mit Hilfe einer anderen Klasse aus einer Textdatei ausgelesen.

3.8.2 Beschreibung der Konstruktoren und Destruktoren

3.8.2.1 Insert::Insert ()

Konstruktor der Klasse. Hier werden die Verbindungsdaten ermittelt.

3.8.3 Dokumentation der Elementfunktionen

3.8.3.1 int Insert::insertIntoDB (string * anw)

Verbindung aufbauen und MySQL-String an die Datenbank schicken. Diese führt ihn dann aus.

Parameter

<i>anw</i>	ist ein Zeiger auf einen in DBPacketInsert aufgebauten SQL-String, der an diese Funktion übergeben wird.
------------	--

Rückgabe

Integerwert, der aber nicht verwendet wird.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Insert.h
- Insert.cpp

3.9 Location Klassenreferenz

```
#include <Location.h>
```

Öffentliche Methoden

- [Location](#) (std::string address, short port)
- std::string [getAddress](#) ()
- int [getPort](#) ()

3.9.1 Ausführliche Beschreibung

Datenstruktur, die Netzwerkdaten bestimmter Teilnehmer speichert.

3.9.2 Beschreibung der Konstruktoren und Destruktoren

3.9.2.1 Location::Location (std::string address, short port)

Erzeugt einen Teilnehmer.

Parameter

<i>address</i>	IP-Adresse des Teilnehmers.
<i>port</i>	Port-Nummer des Teilnehmers.

3.9.3 Dokumentation der Elementfunktionen

3.9.3.1 std::string Location::getAddress ()

Rückgabe

Gibt die Adresse zurück.

3.9.3.2 int Location::getPort ()

Rückgabe

Gibt die Portnummer zurück.

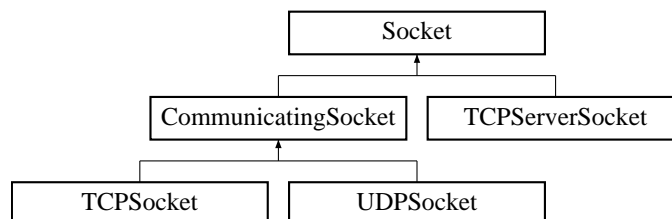
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Location.h
- Location.cpp

3.10 Socket Klassenreferenz

```
#include <PracticalSocket.h>
```

Klassendiagramm für Socket:



Öffentliche Methoden

- `~Socket ()`
- `string getLocalAddress () throw (SocketException)`
- `unsigned short getLocalPort () throw (SocketException)`
- `void setLocalPort (unsigned short localPort) throw (SocketException)`
- `void setLocalAddressAndPort (const string &localAddress, unsigned short localPort=0) throw (SocketException)`

Öffentliche, statische Methoden

- `static unsigned short resolveService (const string &service, const string &protocol="tcp")`

Geschützte Methoden

- `Socket (int type, int protocol) throw (SocketException)`
- `Socket (int sockDesc)`

Geschützte Attribute

- int **sockDesc**

3.10.1 Ausführliche Beschreibung

Basisklasse, welche den Endpunkt der grundlegenden Kommunikation darstellt.

3.10.2 Beschreibung der Konstruktoren und Destruktoren

3.10.2.1 Socket::~~Socket ()

Schließt und dealloziert diesen [Socket](#).

3.10.3 Dokumentation der Elementfunktionen

3.10.3.1 string Socket::getLocalAddress () throw SocketException)

Holt die lokale Adresse.

Rückgabe

local Adresse des Sockets.

Ausnahmebehandlung

<i>Wirft</i>	SocketException falls das Holen fehlschlägt.
--------------	--

3.10.3.2 unsigned short Socket::getLocalPort () throw SocketException)

Holt den lokalen Port.

Rückgabe

Lokalen Port des Sockets.

Ausnahmebehandlung

SocketException	wird geworfen falls das Holen fehlschlägt.
---------------------------------	--

3.10.3.3 unsigned short Socket::resolveService (const string & service, const string & protocol = "tcp") [static]

Ermittelt den spezifizierten Service für das angegebene Protokoll.

Parameter

<i>service</i>	Zu ermittelnde Service (e.g., "http").
<i>protocol</i>	Protokoll des zu ermittelnden Service. Standard ist "tcp".

3.10.3.4 void Socket::setLocalAddressAndPort (const string & localAddress, unsigned short localPort = 0) throw SocketException)

Setzt den lokalen Port und die lokale Adresse. Wenn kein Port angegeben wird, wird ein zufälliger gewählt.

Parameter

<i>localAddress</i>	Lokale Adresse.
<i>localPort</i>	Lokaler Port.

Ausnahmebehandlung

SocketException	wird geworfen falls das Setzen des lokalen Ports oder der Adresse fehlschlägt.
---------------------------------	--

3.10.3.5 void Socket::setLocalPort (unsigned short localPort) throw SocketException)

Setzt den lokalen Port zu einem gewählten Port und die lokale Adresse zu einer beliebigen Schnittstelle.

Parameter

<i>localPort</i>	Lokaler Port
------------------	--------------

Ausnahmebehandlung

SocketException	wird geworfen wenn das Setzen des lokalen Ports fehlschlägt.
---------------------------------	--

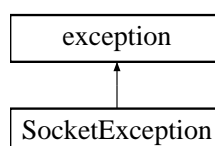
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PracticalSocket.h
- PracticalSocket.cpp

3.11 SocketException Klassenreferenz

```
#include <PracticalSocket.h>
```

Klassendiagramm für SocketException:



Öffentliche Methoden

- `SocketException` (const string &message, bool inclSysMsg=false) throw ()
- `~SocketException` () throw ()
- const char * `what` () const throw ()

3.11.1 Ausführliche Beschreibung

Zeigt ein Problem mit der Ausführung eines Socketaufrufs an.

3.11.2 Beschreibung der Konstruktoren und Destruktoren

3.11.2.1 `SocketException::SocketException (const string & message, bool inclSysMsg = false) throw ()`

Erzeugt eine "SocketException" mit einem erklärenden Hinweistext.

Parameter

<i>message</i>	Nachricht die den Fehlern beschreibt
<i>inclSysMsg</i>	true falls eine Systemnachricht (von strerror(errno)) sollte zu der für den User bereitgestellten Nachricht hinzugefügt werden.

3.11.2.2 `SocketException::~~SocketException () throw ()`

Nur bereitgestellt um zu gewährleisten, dass keine Exceptions zurückgeworfen / zurückgegeben werden.

3.11.3 Dokumentation der Elementfunktionen

3.11.3.1 `const char * SocketException::what () const throw ()`

Holt die Exception-Nachricht.

Rückgabe

Exception-Nachricht.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PracticalSocket.h
- PracticalSocket.cpp

3.12 T_nuex Strukturreferenz

Öffentliche Attribute

- short int **testen** [401]

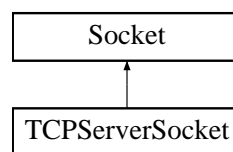
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- mab.cpp

3.13 TCPServerSocket Klassenreferenz

```
#include <PracticalSocket.h>
```

Klassendiagramm für TCPServerSocket:



Öffentliche Methoden

- [TCPServerSocket](#) (unsigned short localPort, int queueLen=5) throw (SocketException)
- [TCPServerSocket](#) (const string &localAddress, unsigned short localPort, int queueLen=5) throw (SocketException)
- [TCPSocket](#) * [accept](#) () throw (SocketException)

Weitere Geerbte Elemente

3.13.1 Ausführliche Beschreibung

TCP-Socket Klasse für Server.

3.13.2 Beschreibung der Konstruktoren und Destruktoren

3.13.2.1 TCPServerSocket::TCPServerSocket (unsigned short *localPort*, int *queueLen* = 5) throw SocketException)

Erzeugt einen TCP-Socket für die Nutzung mit dem Server, welcher zu einer beliebigen Schnittstelle auf dem vereinbarten Port Verbindungen zulässt.

Parameter

<i>localPort</i>	Lokaler Port des Server.
<i>queueLen</i>	Maximale Warteschlangenlänge für ausstehende Verbindungsanfragen. (default 5)

Ausnahmebehandlung

<i>SocketException</i>	wird geworfen falls es nicht möglich ist einen Socket zu erzeugen.
--	--

3.13.2.2 TCPServerSocket::TCPServerSocket (const string & localAddress, unsigned short localPort, int queueLen = 5) throw SocketException)

Erzeugt einen TCP-Socket für die Nutzung mit dem Server, welcher zu einer beliebigen Schnittstelle zu einer vereinbarten Adresse Verbindungen zulässt.

Parameter

<i>localAddress</i>	Lokales Interface (Adresse) des Server-Sockets.
<i>localPort</i>	Lokaler Port des Servers.
<i>queueLen</i>	Maximale Warteschlangenlänge für ausstehende Verbindungsanfragen. (default 5)

Ausnahmebehandlung

<i>SocketException</i>	wird geworfen, falls es nicht möglich ist einen Socket zu erzeugen.
--	---

3.13.3 Dokumentation der Elementfunktionen

3.13.3.1 TCPSocket * TCPServerSocket::accept () throw SocketException)

Blockiert solange bis eine neue Verbindung auf diesem [Socket](#) etabliert wurde oder ein Fehler auftritt.

Rückgabe

Neue Socketverbindung

Ausnahmebehandlung

<i>SocketException</i>	wird geworfen falls der Versuch eine neue Verbindung zu erzeugen fehlschlägt.
--	---

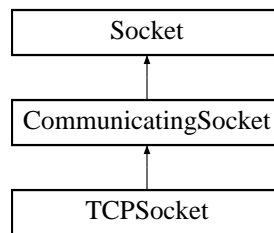
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PracticalSocket.h
- PracticalSocket.cpp

3.14 TCPSocket Klassenreferenz

```
#include <PracticalSocket.h>
```

Klassendiagramm für TCPSocket:



Öffentliche Methoden

- [TCPSocket](#) () throw (SocketException)
- [TCPSocket](#) (const string &foreignAddress, unsigned short foreignPort) throw (SocketException)

Freundbeziehungen

- class **TCPServerSocket**

Weitere Geerbte Elemente

3.14.1 Ausführliche Beschreibung

TCP-Socket für die Kommunikation mit anderen TCP-Sockets.

3.14.2 Beschreibung der Konstruktoren und Destruktoren

3.14.2.1 TCPSocket::TCPSocket () throw SocketException)

Erzeugt einen TCP-Socket mit keiner Verbindung.

Ausnahmebehandlung

SocketException	wird geworfen falls die Erzeugung fehlschlägt.
---------------------------------	--

3.14.2.2 TCPSocket::TCPSocket (const string & foreignAddress, unsigned short foreignPort) throw SocketException)

Erzeugt einen TCP-Socket mit einer Verbindung zu einer bestimmten Adresse und einem bestimmten Port.

Parameter

<i>foreign-Address</i>	foreign address (IP address or name)
<i>foreignPort</i>	foreign port

Ausnahmebehandlung

SocketException	wird geworfen falls die Erzeugung fehlschlägt.
---------------------------------	--

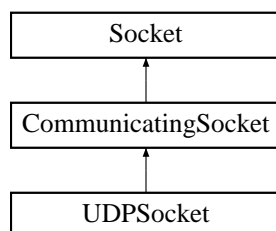
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PracticalSocket.h
- PracticalSocket.cpp

3.15 UDPSocket Klassenreferenz

```
#include <PracticalSocket.h>
```

Klassendiagramm für UDPSocket:



Öffentliche Methoden

- [UDPSocket](#) () throw (SocketException)
- [UDPSocket](#) (unsigned short localPort) throw (SocketException)
- [UDPSocket](#) (const string &localAddress, unsigned short localPort) throw (SocketException)
- void [disconnect](#) () throw (SocketException)
- void [sendTo](#) (const void *buffer, int bufferLen, const string &foreignAddress, unsigned short foreignPort) throw (SocketException)
- int [recvFrom](#) (void *buffer, int bufferLen, string &sourceAddress, unsigned short &sourcePort) throw (SocketException)
- void [setMulticastTTL](#) (unsigned char multicastTTL) throw (SocketException)
- void [joinGroup](#) (const string &multicastGroup) throw (SocketException)
- void [leaveGroup](#) (const string &multicastGroup) throw (SocketException)

Weitere Geerbte Elemente

3.15.1 Ausführliche Beschreibung

UDP-Socket Klasse

3.15.2 Beschreibung der Konstruktoren und Destruktoren

3.15.2.1 UDPSocket::UDPSocket () throw SocketException)

Erzeugt einen UDP-Socket.

Ausnahmebehandlung

SocketException	wird geworfen, falls die Erzeugung fehlschlägt.
---------------------------------	---

3.15.2.2 UDPSocket::UDPSocket (unsigned short *localPort*) throw SocketException)

Erzeugt einen UDP-Socket mit einem spezifischen Port.

Parameter

<i>localPort</i>	Lokaler Port
------------------	--------------

Ausnahmebehandlung

SocketException	wird geworfen, falls die Erzeugung fehlschlägt.
---------------------------------	---

3.15.2.3 UDPSocket::UDPSocket (const string & *localAddress*, unsigned short *localPort*) throw SocketException)

Erzeugt einen UDP-Socket mit einer spezifischen Adresse und einem gegebenen Port.

Parameter

<i>localAddress</i>	Lokale Adresse.
<i>localPort</i>	Lokaler Port.

Ausnahmebehandlung

SocketException	wird geworfen, falls die Erzeugung fehlschlägt.
---------------------------------	---

3.15.3 Dokumentation der Elementfunktionen

3.15.3.1 void UDPSocket::disconnect () throw SocketException)

Setze Adresse und Port zurück.

Rückgabe

true falls kein Fehler auftrat.

Ausnahmebehandlung

<i>SocketException</i>	wird geworfen, falls eine Trennung fehlschlägt.
--	---

3.15.3.2 void UDPSocket::joinGroup (const string & *multicastGroup*) throw **SocketException**)

Tritt der angegebenen Multicast-Gruppe bei.

Parameter

<i>multicast-Group</i>	Adresse der Multicast-Gruppe.
------------------------	-------------------------------

Ausnahmebehandlung

<i>SocketException</i>	wird geworfen, falls das Beitreten fehlschlägt.
--	---

3.15.3.3 void UDPSocket::leaveGroup (const string & *multicastGroup*) throw **SocketException**)

Verlasse die angegebene Multicast-Gruppe.

Parameter

<i>multicast-Group</i>	Zu verlassende Multicast-Gruppe.
------------------------	----------------------------------

Ausnahmebehandlung

<i>SocketException</i>	wird geworfen, falls das Verlassen fehlschlägt.
--	---

3.15.3.4 int UDPSocket::recvFrom (void * *buffer*, int *bufferLen*, string & *sourceAddress*, unsigned short & *sourcePort*) throw **SocketException**)

Empfängt eine Nachricht an einem UDP-Socket.

Parameter

<i>buffer</i>	Buffer, in den gelesen werden soll.
<i>bufferLen</i>	Maximale Anzahl an Bytes, die empfangen werden sollen.
<i>source-Address</i>	Adresse, von der die Nachricht stammt.
<i>sourcePort</i>	Portnummer des Senders.

Rückgabe

Anzahl der Bytes die empfangen wurden, -1 falls ein Fehler auftrat.

Ausnahmebehandlung

SocketException	wird geworfen, falls das Empfangen fehlschlägt.
---------------------------------	---

3.15.3.5 void UDPSocket::sendTo (const void * *buffer*, int *bufferLen*, const string & *foreignAddress*, unsigned short *foreignPort*) throw SocketException)

Sendet einen Buffer als UDP-Datagramm an eine bestimmte Adresse und Portnummer.

Parameter

<i>buffer</i>	Buffer, der gesendet werden soll.
<i>bufferLen</i>	Anzahl der Bytes, die geschrieben werden sollen.
<i>foreign-Address</i>	Adresse, an die versendet werden soll.
<i>foreignPort</i>	Portnummer, an die versendet werden soll.

Rückgabe

true falls Versand geklappt hat.

Ausnahmebehandlung

SocketException	wird geworfen falls, das Senden fehlschlägt.
---------------------------------	--

3.15.3.6 void UDPSocket::setMulticastTTL (unsigned char *multicastTTL*) throw SocketException)

Setzt Multicast TTL.

Parameter

<i>multicastTTL</i>	Multicast TTL.
---------------------	----------------

Ausnahmebehandlung

SocketException	wird geworfen, falls das Setzen fehlschlägt.
---------------------------------	--

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PracticalSocket.h
- PracticalSocket.cpp

Index

- ~Socket
 - Socket, [18](#)
- ~SocketException
 - SocketException, [20](#)
- accept
 - TCPServerSocket, [22](#)
- CommunicatingSocket, [3](#)
 - connect, [4](#)
 - getForeignAddress, [4](#)
 - getForeignPort, [4](#)
 - recv, [4](#)
 - send, [5](#)
- connect
 - CommunicatingSocket, [4](#)
- DBPacketInsert, [9](#)
 - db_insert, [10](#)
- Data, [5](#)
 - Data, [6](#)
 - getDatatype, [6](#)
 - getPosition, [6](#)
 - getValue, [6](#)
- DatatypeDaemon, [6](#)
 - DatatypeDaemon, [7](#)
 - DatatypeDaemon, [7](#)
 - getPosActualPacket, [7](#)
 - getTime, [7](#)
 - parseNextValue, [7](#)
- db_insert
 - DBPacketInsert, [10](#)
- dbData, [8](#)
 - dbData, [8](#)
 - dbData, [8](#)
 - getDB, [9](#)
 - getHost, [9](#)
 - getPW, [9](#)
 - getUser, [9](#)
- Decoder, [10](#)
 - Decoder, [10](#), [12](#)
 - getNextData, [12](#)
 - getPackageNum, [12](#)
 - getPackagePos, [12](#)
- disconnect
 - UDPSocket, [25](#)
- Encoder, [13](#)
 - Encoder, [13](#)
 - getNextPackage, [14](#)
 - getPackage, [14](#)
 - getPackageSize, [14](#)
 - getPackageSum, [14](#)
- getAddress
 - Location, [16](#)
- getDB
 - dbData, [9](#)
- getDatatype
 - Data, [6](#)
- getForeignAddress
 - CommunicatingSocket, [4](#)
- getForeignPort
 - CommunicatingSocket, [4](#)
- getHost
 - dbData, [9](#)
- getLocalAddress
 - Socket, [18](#)
- getLocalPort
 - Socket, [18](#)
- getNextData
 - Decoder, [12](#)
- getNextPackage
 - Encoder, [14](#)
- getPW
 - dbData, [9](#)
- getPackage
 - Encoder, [14](#)
- getPackageNum
 - Decoder, [12](#)
- getPackagePos
 - Decoder, [12](#)

- getPackageSize
 - Encoder, 14
- getPackageSum
 - Encoder, 14
- getPort
 - Location, 16
- getPosActualPacket
 - DatatypeDaemon, 7
- getPosition
 - Data, 6
- getTime
 - DatatypeDaemon, 7
- getUser
 - dbData, 9
- getValue
 - Data, 6
- Insert, 15
 - Insert, 15
 - insertIntoDB, 15
- insertIntoDB
 - Insert, 15
- joinGroup
 - UDPSocket, 26
- leaveGroup
 - UDPSocket, 26
- Location, 16
 - getAddress, 16
 - getPort, 16
 - Location, 16
- parseNextValue
 - DatatypeDaemon, 7
- recv
 - CommunicatingSocket, 4
- recvFrom
 - UDPSocket, 26
- resolveService
 - Socket, 18
- send
 - CommunicatingSocket, 5
- sendTo
 - UDPSocket, 27
- setLocalAddressAndPort
 - Socket, 19
- setLocalPort
 - Socket, 19
- setMulticastTTL
 - UDPSocket, 27
- Socket, 17
 - ~Socket, 18
 - getLocalAddress, 18
 - getLocalPort, 18
 - resolveService, 18
 - setLocalAddressAndPort, 19
 - setLocalPort, 19
- SocketException, 19
 - ~SocketException, 20
 - SocketException, 20
 - SocketException, 20
 - what, 20
- T_nuex, 20
- TCPServerSocket, 21
 - accept, 22
 - TCPServerSocket, 21, 22
 - TCPServerSocket, 21, 22
- TCPSocket, 22
 - TCPSocket, 23
 - TCPSocket, 23
- UDPSocket, 24
 - disconnect, 25
 - joinGroup, 26
 - leaveGroup, 26
 - recvFrom, 26
 - sendTo, 27
 - setMulticastTTL, 27
 - UDPSocket, 25
 - UDPSocket, 25
- what
 - SocketException, 20