



—

Simulationsumgebung für digitale und analoge Ein- und Ausgänge

—

Benutzerdokumentation

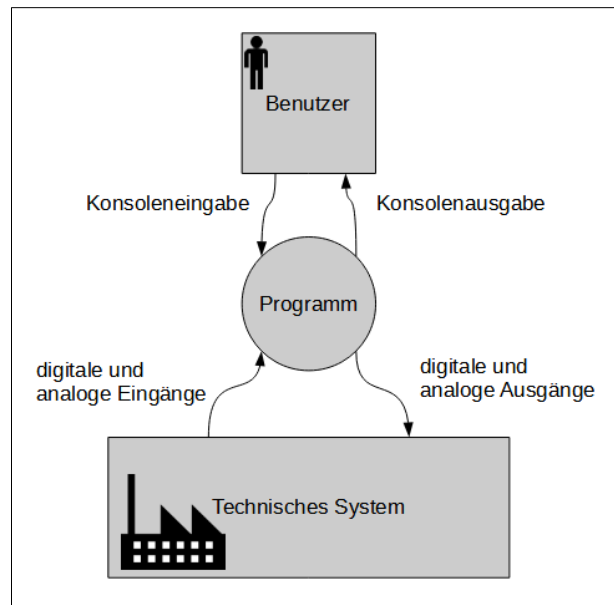
Autor	Markus Breuer
Version	0.5
Datum	13.02.2025

Inhaltsverzeichnis

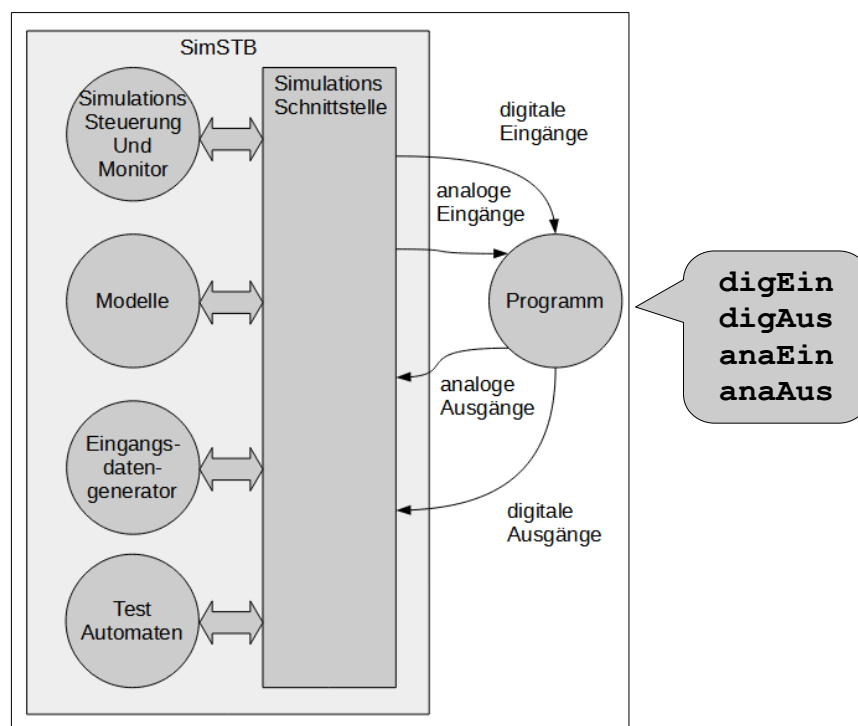
1 Allgemeines.....	2
2 Lokale Installation der Simulationsumgebung.....	4
2.1 Voraussetzungen.....	4
2.2 Installation.....	4
2.3 Quelldateien.....	5
3 Steuerung der Simulationsumgebung SimSTB.....	6
4 Erstellung eigener Programme für die Simulationsumgebung SimSTB.....	9
4.1 C/C++ - Schnittstelle.....	9
4.2 Python-Schnittstelle.....	12
5 SimSTB Ein- und Ausgangsbelegung.....	15

1 Allgemeines

Oft muss ein Programm nicht nur über die Konsole oder eine graphische Benutzeroberfläche mit dem Benutzer kommunizieren, sondern auch über analoge und digitale Schnittstellen mit einem technischen System.



Die Simulationsumgebung **SimSTB** erlaubt es, dies für Schulungszwecke auch ohne zusätzliche Hardware mittels Simulation durchzuführen.



SimSTB besteht aus zwei Teilen:

1. Der eigentlichen **Simulationsumgebung** und **Steuerprogrammen** für diese. Die Steuerprogramme sind in Abschnitt 3 beschrieben. Die Installation in Abschnitt 2.
2. Einer **Programmier-Schnittstelle**, um aus eigenen Programmen die Simulationsumgebung zu nutzen. Es sind eine **C/C++-Schnittstelle** und eine **Python-Schnittstelle** vorhanden. In Abschnitt 4 ist beschrieben, wie Sie eigene Programme erstellen können.

Um die Simulationsumgebung SimSTB aus eigenen Programmen zu nutzen, stehen 4 einfache Funktionen zur Verfügung.

Schnittstelle	Funktion	Kanäle/id	Typ	Richtung
Digitaler Eingang	digEin	0 .. 15	digital	Eingang
Digitaler Ausgang	digAus	0 .. 15	digital	Ausgang
Analoger Eingang	anaEin	0 .. 7	analog	Eingang
Analoger Ausgang	anaAus	0 .. 7	analog	Ausgang

Die Kanäle bestimmen die Anzahl der jeweiligen Schnittstellen.

2 Lokale Installation der Simulationsumgebung

2.1 Voraussetzungen

- keine

2.2 Installation

- Kopieren Sie das bereitgestellte Simulationsverzeichnis (in GitHub [/auslieferung/sim](#)) samt Unterverzeichnissen nach „C:\“. Solange Sie nur die Simulationsumgebung nutzen wollen, und keine Modifikationen an deren Quellcode vornehmen wollen brauchen Sie keine weiteren Dateien.
- Kontrollieren Sie, ob folgende Verzeichnis-Struktur und Dateien vorhanden sind.

```
C:\SIM
|
|   anaaus.txt
|   anaein.txt
|   digaus.txt
|   digein.txt
|   LICENSE.txt
|   README.md
|   simstb.log
|
|---beispiele
|   |   beispiel.cpp
|   |   beispiel.py
|
|---bin
|   |   band_links.gif
|   |   band_rechts.gif
|   |   band_stopp.gif
|   |   led_blau.png
|   |   led_grau.png
|   |   led_gruen.png
|   |   modelle.json
|   |   simstb.ico
|   |   simstb_gui.exe
|   |   simstb_modell_1.exe
|   |   simstb_modell_2.exe
|
|---dokumentation
|   |   SimSTB-Benutzerdokumentation.pdf
|
|---header
|   |   simulation.h
|
|---lib
|   |   simlib.lib
```

```
python
  simstb_dateizugriff.py
  simstb_konfig.py
  simstb_logger.py
  simulator.py
```

2.3 Quelldateien

Bei der Simulationsumgebung SimSTB handelt es sich um Open Source Software. Diese steht über GitHub (<https://markusbreuer1964.github.io/SimSTB/>) zur Verfügung.

3 Steuerung der Simulationsumgebung SimSTB

Im Unterverzeichnis bin finden Sie Programme zur Bedienung der Simulationsumgebung:

1. Simulations Steuerung und Monitoring

Mit Hilfe des Programms `simstb_gui.exe` können Sie digitalen und analogen Ein- und Ausgänge überwachen und die Eingänge setzen. Die Werte werden im Sekundentakt aktualisiert. Starten können Sie das Programm über einen einfachen Doppelklick auf die Exe-Datei.

SimSTB
Simulationsumgebung für digitale und analoge Ein- und Ausgänge

03.08.2021 10:29:18

Alles 0
Alle Ausgänge 0
Alle Eingänge 0
Digitale Eingänge 0
Digitale Eingänge 1

Modelle
Testautomaten
Funktionsgenerator
Datenaufzeichnung

Digitale Eingänge

- ☐ DE0
- ☐ DE1
- ☐ DE2
- ☐ DE3
- ☐ DE4
- ☐ DE5
- ☐ DE6
- ☐ DE7
- ☐ DE8
- ☐ DE9
- ☐ DE10
- ☐ DE11
- ☐ DE12
- ☐ DE13
- ☐ DE14
- ☐ DE15

Digitale Ausgänge

- ☐ DA0
- ☐ DA1
- ☐ DA2
- ☐ DA3
- ☐ DA4
- ☐ DA5
- ☐ DA6
- ☐ DA7
- ☐ DA8
- ☐ DA9
- ☐ DA10
- ☐ DA11
- ☐ DA12
- ☐ DA13
- ☐ DA14
- ☐ DA15

Analoge Eingänge

AE0	80.0
AE1	0.0
AE2	0.0
AE3	0.0
AE4	0.0
AE5	0.0
AE6	0.0
AE7	0.0

Analoge Ausgänge

AA0	0.0
AA1	0.0
AA2	0.0
AA3	0.0
AA4	0.0
AA5	0.0
AA6	0.0
AA7	0.0

Beenden

- Durch Mausklick auf die digitalen Eingänge können Sie bei der Eingangsbelegung zwischen 0 und 1 wechseln.
- Nach Auswahl eines analogen Eingangs können Sie dort einen analogen Zahlenwert eingeben. Als Dezimaltrennzeichen müssen Sie einen Punkt benutzen.
- Mit Hilfe der nebenstehenden Knöpfe können Sie ganze Gruppen von Ein- bzw. Ausgängen zusammen setzen.



- Mit Hilfe es Knopfs Datenaufzeichnung können Sie eine Datenaufzeichnung starten.



Mit Hilfe der Knöpfe **Starten** und **Stoppen** kontrollieren Sie die Datenaufzeichnung. Solange die Aufzeichnung läuft werden jeweils alle digitalen und analogen Ein- und Ausgabewerte in eine CSV-Datei geschrieben

- Mit Hilfe des Knopfs Funktionsgenerator können Sie einen analogen Funktionsgenerator starten.

Funktionsgenerator

Kanal	An/Aus	Signalform	Amplitude	P.Dauer (s)
AE0	<input type="checkbox"/>	Zufall	0	0
AE1	<input type="checkbox"/>	Zufall	0	0
AE2	<input type="checkbox"/>	Zufall	0	0
AE3	<input type="checkbox"/>	Zufall	0	0
AE4	<input type="checkbox"/>	Zufall	0	0
AE5	<input type="checkbox"/>	Zufall	0	0
AE6	<input type="checkbox"/>	Zufall	0	0
AE7	<input type="checkbox"/>	Zufall	0	0

Starten Stoppen Generator nicht aktiv

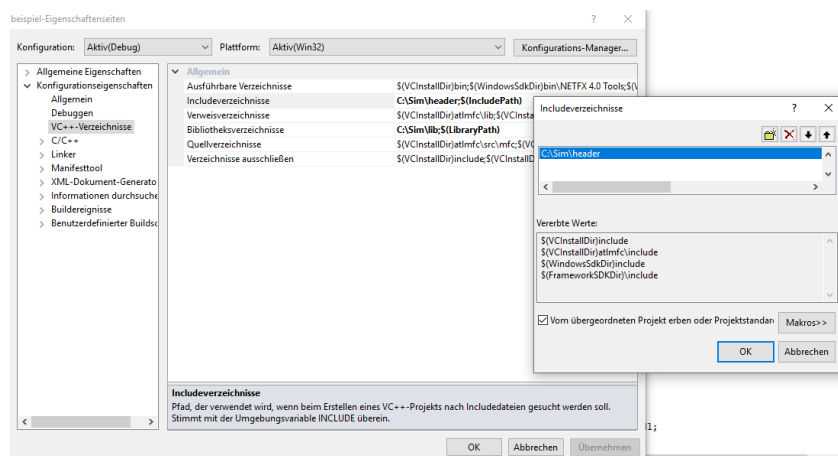
Schließen

Hier können Sie einzelnen analogen Eingangskanäle aktivieren, die jeweilige Signalform auswählen und Parameter zur Signalform einstellen. Mit Hilfe der Knöpfe **Starten** und **Stoppen** kontrollieren Sie den Funktionsgenerator.

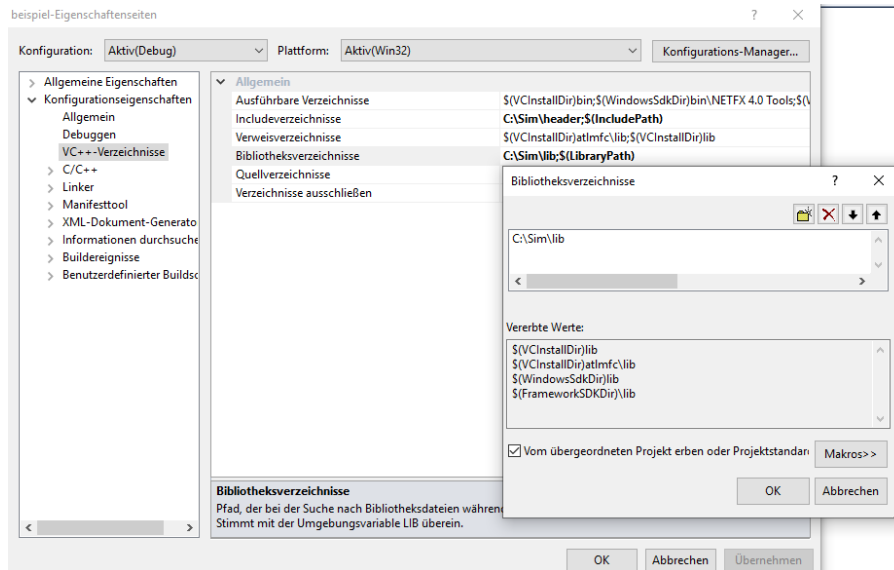
4 Erstellung eigener Programme für die Simulationsumgebung SimSTB

4.1 C/C++ - Schnittstelle

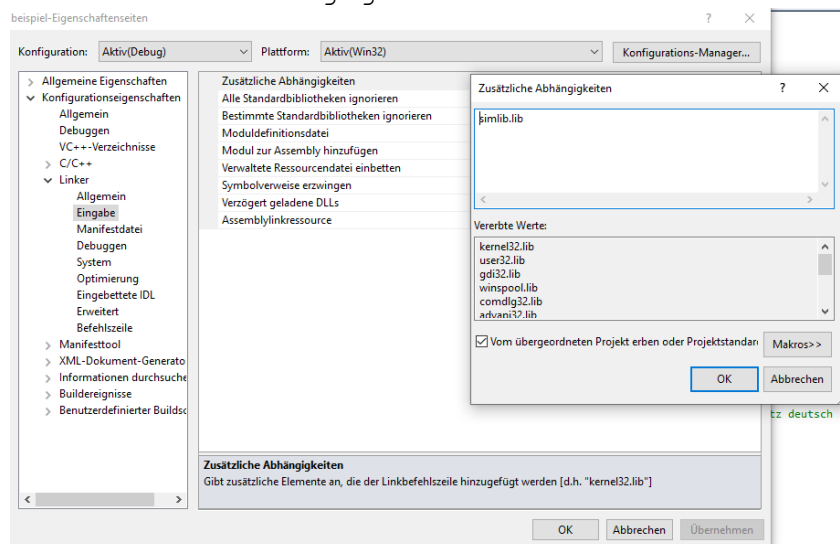
- a) Um die Simulationsumgebung SimSTB nutzen zu können, müssen 4 Einstellungen vorgenommen werden:
1. Der **Suchpfad für Header-Dateien muss erweitert werden**. Hier muss der Pfad `C : \ Sim\header` ergänzt werden. Vorgenommen werden die Einstellungen unter Projekt → Eigenschaften → VC++Verzeichnisse → Includeverzeichnisse → Bearbeiten.



2. Die **Header-Datei `simulation.h` muss in der eigenen CPP-Datei inkludiert werden**. Dabei sind Anführungszeichen und keine spitzen Klammern zu verwenden. Im Beispiel-Code unten sieht man ein Beispiel hierfür.
3. Der **Suchpfad für Bibliotheken muss erweitert werden**. Hier muss der Pfad `C : \ Sim\lib` ergänzt werden. Vorgenommen werden die Einstellungen unter Projekt → Eigenschaften → VC++Verzeichnisse → Includeverzeichnisse → Bearbeiten.



4. Die **Bibliothek simlib.lib** muss ergänzt werden. Vorgenommen werden die Einstellungen unter Projekt → Eigenschaften → Linker → Eingabe → Zusätzliche Abhängigkeiten → Bearbeiten.



- b) Danach kann normal weiter programmiert werden, wobei man die vier Funktionen zur Nutzung der Simulationsumgebung benutzen darf.

Bemerkung:

- Bei der Bibliothek **simlib.lib** handelt es sich um eine 32-bit Bibliothek. Diese wurde der Compileroption **/MDd** (Multithreaded-Debug-DLL) erstellt. Falls eine andere Version benötigt wird, kann diese mit Hilfe der Open Source Dateien (siehe Abschnitt 2.3 Quelldateien) selber erstellt werden.

Funktionsprototypen:

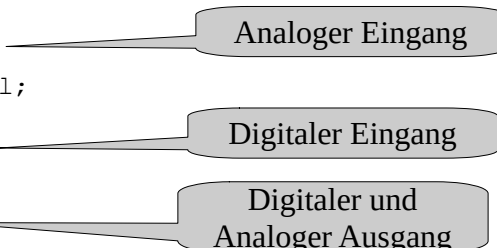
```
const int DIGMAXLAENGE = 16;
const int ANAMAXLAENGE = 8;

bool digEin( int id);
void digAus( int id, bool wert);

double anaEin( int id);
void anaAus( int id, double wert);
```

Beispiel-Code (Auszug; vollständig in Unterordner beispiele):

```
...
#include "simulation.h"
...
int main()
{
    bool ende = false;
    double wert;
    ...
    while( ende != true)
    {
        wert = anaEin( 0);
        cout << wert << endl;
        Sleep( 1000);
        ende = digEin( 0);
    }
    digAus( 15, 1);
    anaAus( 7, -123.456);
    ...
}
```



4.2 Python-Schnittstelle

Benötigte Python Pakete:

Damit die Python-Schnittstelle genutzt werden kann, muss das Paket filelock installiert sein.

```
Pip install filelock
```

Voraussetzungen:

Damit das eigene Programm die Schnittstellenfunktionen nutzen kann muss man die Datei `simulator.py` importieren. Hierzu dient die folgende Anweisung:

```
import simulator as sim
```

Damit das Modul `simulator.py` gefunden wird, muss der Python-Modul-Pfad ergänzt werden. Dies geschieht durch Anlegen bzw. Erweitern der Umgebungsvariablen PYTHONPATH. Die Umgebungsvariable muss das Verzeichnis `C:\Sim\python` enthalten.

Path	C:\Users\marku\meine_exen;C:\Program Files\Python39\Scripts\;C:...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	3c03
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\sys...
PYTHONPATH	C:\Sim\python
STADTNAME	Strauch

Schnittstellenfunktionen:

```
DIGMAXLAENGE = 16
ANAMAXLAENGE = 8

def dig_ein( id)
def dig_aus( id, wert)

def ana_ein( id)
def ana_aus( id, wert)
```

- Der Parameter `id` ist vom Datentyp `int`.
- `id` ist zwischen 0 und 15 bei den beiden Funktionen für die digitale Ein- und Ausgabe.
- `id` ist zwischen 0 und 7 bei den beiden Funktionen für die analoge Ein- und Ausgabe.
- Der Parameter `wert` ist vom Datentyp `bool` bei der digitalen Ausgabe, vom Datentyp `float` bei der analogen Ausgabe.
- Die Funktion `dig_ein` gibt einen Wert vom Datentyp `bool` zurück. Die Funktion `ana_ein` gibt einen Wert vom Datentyp `float` zurück.
- Wenn ein ungültiger Index genutzt wird, geben `dig_ein` und `ana_ein` `None` zurück.
- Wenn ein ungültiger Index genutzt wird, speichern `dig_aus` und `ana_aus` nichts ab.

Log-Datei:

- Im Wurzelverzeichnis kann sich eine Log-Datei namens `simstb.log` befinden, welche Hinweise über aufgetretene Fehlsituationen gibt

Beispiel-Code (Auszug; vollständig in Unterordner `beispiele`):

```
...
import simulator as sim
...
def test():
    """ Testfunktion """
    ende = False
    ...
    while ende != True:
        wert = sim.ana_ein( 0)
        print(wert)
        time.sleep( 1)
        ende = sim.dig_ein( 0)

    sim.dig_aus( 15, 1)
    sim.ana_aus( 7, -123.456)
    ...


test()
```

Analoger Eingang einlesen

Digitaler Eingang einlesen

Digitaler und
Analoger Ausgang setzen

5 SimSTB Ein- und Ausgangsbelegung

SinSTB Ein- und Ausgangsbelegung 		
Digital Eingänge		
	<input type="radio"/> DE0	
	<input type="radio"/> DE1	
	<input type="radio"/> DE2	
	<input type="radio"/> DE3	
	<input type="radio"/> DE4	
	<input type="radio"/> DE5	
	<input type="radio"/> DE6	
	<input type="radio"/> DE7	
	<input type="radio"/> DE8	
	<input type="radio"/> DE9	
	<input type="radio"/> DE10	
	<input type="radio"/> DE11	
	<input type="radio"/> DE12	
	<input type="radio"/> DE13	
	<input type="radio"/> DE14	
	<input type="radio"/> DE15	
Analoge Eingänge		
	<input type="radio"/> AE0	
	<input type="radio"/> AE1	
	<input type="radio"/> AE2	
	<input type="radio"/> AE3	
	<input type="radio"/> AE4	
	<input type="radio"/> AE5	
	<input type="radio"/> AE6	
	<input type="radio"/> AE7	
Digitale Ausgänge		
DA0	<input type="radio"/>	
DA1	<input type="radio"/>	
DA2	<input type="radio"/>	
DA3	<input type="radio"/>	
DA4	<input type="radio"/>	
DA5	<input type="radio"/>	
DA6	<input type="radio"/>	
DA7	<input type="radio"/>	
DA8	<input type="radio"/>	
DA9	<input type="radio"/>	
DA10	<input type="radio"/>	
DA11	<input type="radio"/>	
DA12	<input type="radio"/>	
DA13	<input type="radio"/>	
DA14	<input type="radio"/>	
DA15	<input type="radio"/>	
Analoge Ausgänge		
AA0	<input type="radio"/>	
AA1	<input type="radio"/>	
AA2	<input type="radio"/>	
AA3	<input type="radio"/>	
AA4	<input type="radio"/>	
AA5	<input type="radio"/>	
AA6	<input type="radio"/>	
AA7	<input type="radio"/>	