



—

Simulationsumgebung für digitale und analoge Ein- und Ausgänge

—

Benutzerdokumentation

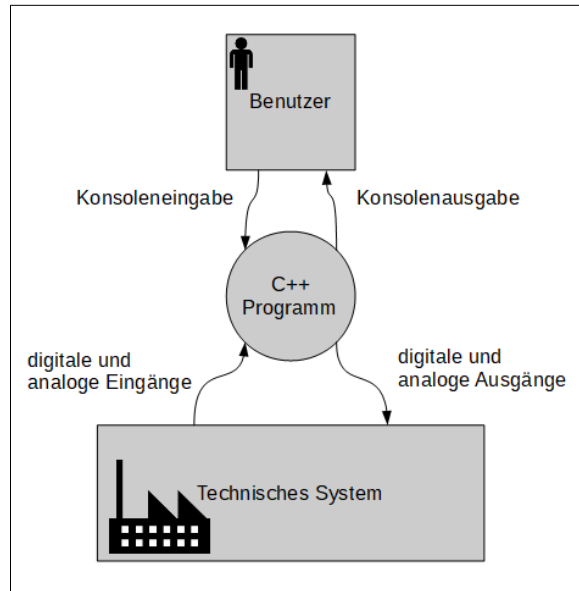
Autor	Markus Breuer
Version	0.3
Datum	06.08.2021

Inhaltsverzeichnis

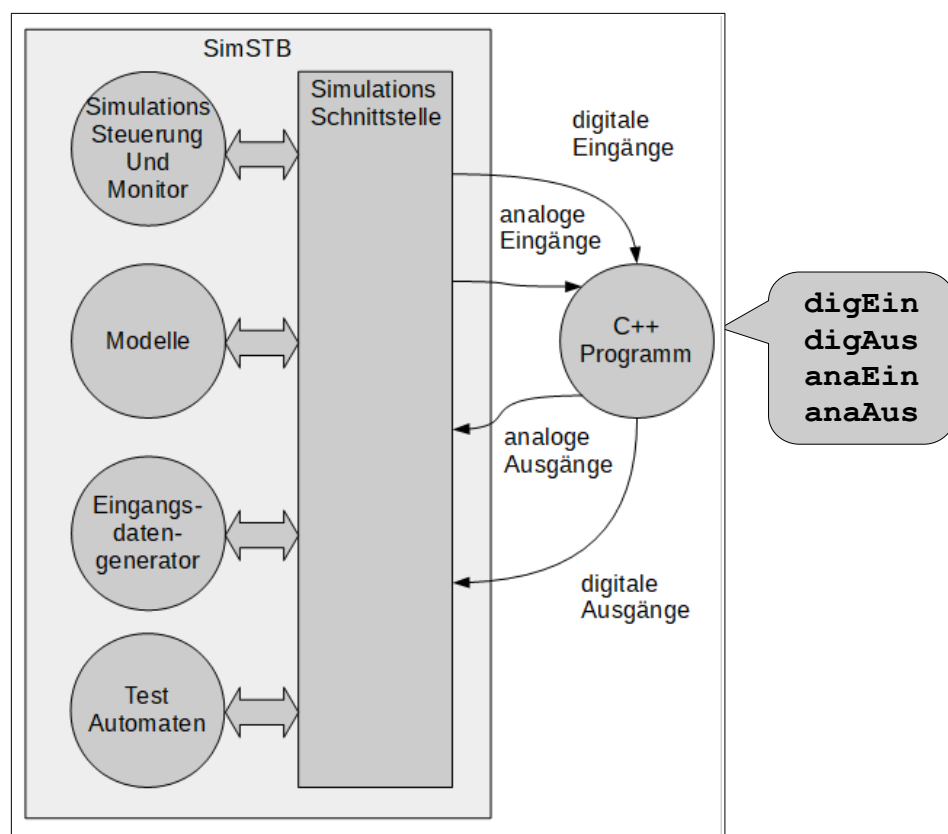
1 Allgemeines.....	2
2 Lokale Installation der Simulationsumgebung.....	4
2.1 Voraussetzungen.....	4
2.2 Installation.....	4
2.3 Quelldateien.....	4
3 Steuerung der Simulationsumgebung SimSTB.....	5
4 Erstellung eigener Programme für die Simulationsumgebung SimSTB.....	8
5 SimSTB Ein- und Ausgangsbelegung.....	11

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. In Abschnitt 4 ist beschrieben, wie Sie eigene Programme erstellen können.

Um die Simulationsumgebung SimSTB aus eigenen Programmen zu nutzen, stehen 4 einfache C++-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
│   ├── beispiele
│   │   └── beispiel.cpp
│   ├── bin
│   │   ├── simstb_gui.exe
│   │   └── sim_stb.ico
│   ├── dokumentation
│   │   └── SimSTB-Benutzerdokumentation.pdf
│   ├── header
│   │   └── simulation.h
│   └── lib
│       └── simlib.lib
```

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		Digitale Ausgänge	
<input type="checkbox"/> DE0		<input type="checkbox"/> DA0	
<input type="checkbox"/> DE1		<input type="checkbox"/> DA1	
<input type="checkbox"/> DE2		<input type="checkbox"/> DA2	
<input type="checkbox"/> DE3		<input type="checkbox"/> DA3	
<input type="checkbox"/> DE4		<input type="checkbox"/> DA4	
<input type="checkbox"/> DE5		<input type="checkbox"/> DA5	
<input type="checkbox"/> DE6		<input type="checkbox"/> DA6	
<input type="checkbox"/> DE7		<input type="checkbox"/> DA7	
<input type="checkbox"/> DE8		<input type="checkbox"/> DA8	
<input type="checkbox"/> DE9		<input type="checkbox"/> DA9	
<input type="checkbox"/> DE10		<input type="checkbox"/> DA10	
<input type="checkbox"/> DE11		<input type="checkbox"/> DA11	
<input type="checkbox"/> DE12		<input type="checkbox"/> DA12	
<input type="checkbox"/> DE13		<input type="checkbox"/> DA13	
<input type="checkbox"/> DE14		<input type="checkbox"/> DA14	
<input type="checkbox"/> DE15		<input type="checkbox"/> DA15	

Analoge Eingänge		Analoge Ausgänge	
AE0	80.0	AA0	0.0
AE1	0.0	AA1	0.0
AE2	0.0	AA2	0.0
AE3	0.0	AA3	0.0
AE4	0.0	AA4	0.0
AE5	0.0	AA5	0.0
AE6	0.0	AA6	0.0
AE7	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

Funktionsgenerator Steuerung

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.

2. Zukünftige Erweiterungen

Mit Hilfe der nebenstehenden Knöpfe sollen geplante Erweiterungen integriert werden.

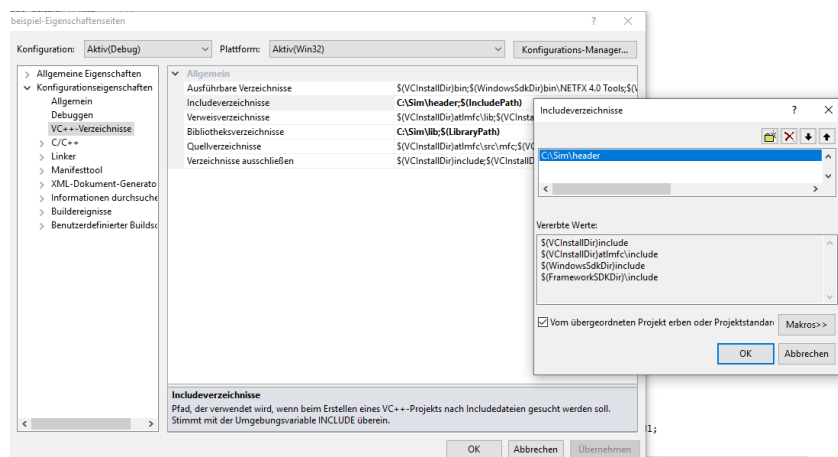
- **Modelle** (R6) sind visuelle Umsetzungen, wie z.B. eine Ampleanlage oder ein Fließband. Diese sollen ein direktes Ausprobieren von Schülerlösungen erlauben.
- Bei **Testautomaten** (R9) handelt es sich um Eingangsvorbelegungen und erwartete Ausgangszustände. Diese werden dann automatisiert ablaufen gelassen und die Soll- und Ist-Zustände der Ausgänge miteinander verglichen.

Modelle

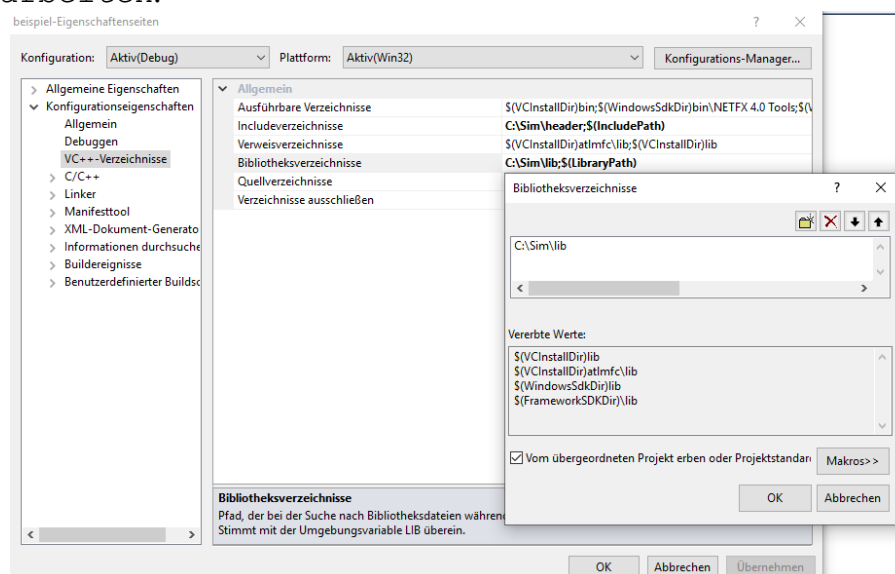
Testautomaten

4 Erstellung eigener Programme für die Simulationsumgebung SimSTB

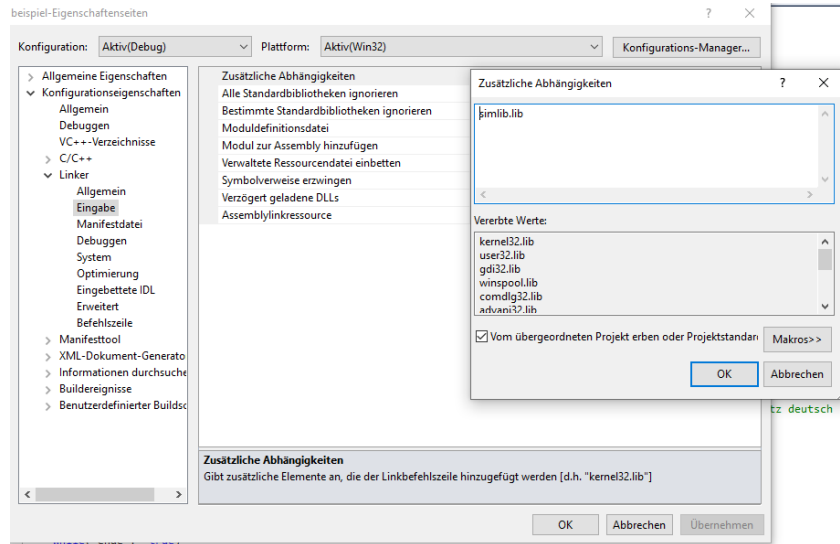
- a) Um die Simulationsumgebung SimSTB nutzen zu können, müssen 4 Einstellungen vorgenommen werden:
1. Der **Suchpfad für Header-Dateien muss erweitert werden**. Hier muss der Pfad `C:\Sim\header` ergänzt werden. Vorgenommen werden die Einstellungen unter Projekt → Eigenschaften → VC++Verzeichnisse →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

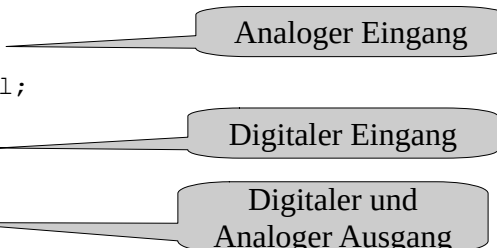
```
const int DIGMAXLAENGE = 16;
const int ANAMAXLAENGE = 8;

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);
    ...
}
```



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"/>	