AnTeS

Entwicklung und Anwendung des Analyse- und Testsystems der GRS

GRS - 648

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

AnTeS

Entwicklung und Anwendung des Analyse- und Testsystems der GRS

Forschungsarbeiten zur Weiterentwicklung der Methode der Sensitivitäts-analyse zur Bewertung von Fehlerauswirkungen auf ein Leittechnik-Testsystem

Christian Müller
Ewgenij Piljugin
Patrick Gebhardt
Jaroslaw Shvab

März 2021

Anmerkung:

Das diesem Bericht zugrunde liegende Forschungsvorhaben wurde mit Mitteln des Bundesministeriums für Umwelt, Naturschutz und nukleare Sicherheit (BMU) unter dem Förderkennzeichen 4718R01314 durchgeführt.

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei der GRS.

Der Bericht gibt die Auffassung und Meinung der GRS wieder und muss nicht mit der Meinung des BMU übereinstimmen.

GRS - 648
ISBN 978-3-949088-37-7

Deskriptoren

Digitale Leittechnik, Sensitivitätsanalyse, Simulation, Testsystem

Inhaltsverzeichnis

1 Einleitung 1

2 AnTeS – das Analyse- und Testsystem der GRS 3

2.1 Modul 1: Reales Leittechniksystem 4

2.2 Modul 2: Simulierte Leittechniksysteme 7

2.3 Modul 3: Anbindung externer Signale 17

2.3.1 AnTeS-Interfaces 17

2.3.2 SimGen – Simulation verfahrenstechnischer Systeme 23

2.3.3 KranSim – Simulation von Maschinenhauskränen 24

2.3.4 TeSys – Testsystem für reale verfahrenstechnische Systeme 25

3 Werkzeug- und Methodenentwicklungen 35

3.1 Fehlerinjektion im AnTeS 35

3.2 Automatisierte Fehlerauswirkungsanalysen im AnTeS 45

3.3 Monte-Carlo-Simulationen im AnTeS 52

4 Durchgeführte Analysen und Tests 53

4.1 Automatisierte Fehlerauswirkungsanalysen mit AnTeS 53

4.2 Sensitivitätsanalysen mit AnTeS 59

4.2.1 Berücksichtigung von sofortigen Reparaturen 62

4.2.2 Berücksichtigung von WKPen 65

4.2.3 Durchführung der Analysen 66

4.2.4 Ergebnisse 68

4.2.5 Bewertung der Ergebnisse 72

4.3 Tests mit verfahrenstechnischen Simulationen 77

4.3.1 Tests mit SimGen 77

4.3.2 Tests mit KranSim 81

5 Zusammenfassung und Ausblick 89

6 Referenzen 93

7 Abbildungsverzeichnis 95

8 Abkürzungen 99

A Anhang 101

A.1 Direkter Zugriff auf das Modul 1 von AnTeS (TXS) im laufenden Betrieb 101

A.2 Erstellung und Verwendung von DLLs im Modul 2 von AnTeS 106

A.3 Modellsysteme und Analysemethodik aus dem Vorgängervorhaben 114

A.4 Detaillierte Beschreibung von SimGen 124

A.5 Detaillierte Beschreibung von KranSim 137

1 Einleitung

Moderne Leittechniksysteme in Kernkraftwerken weltweit basieren zunehmend auf rechnerbasierten und programmierbaren (im Folgenden zusammenfassend: digitale) Einrichtungen[1]. Derartige Systeme sind generell aufgrund komplexerer Architekturen und Hardware weitaus schwieriger auf Fehlerfreiheit zu prüfen als entsprechende festverdrahtete Systeme mit ähnlichen Leittechnik-Funktionen. Latente Programmierfehler, fehlerbehaftete Updates der Software und des Konfigurationsmanagements der Hardware, Lücken in der IT-Sicherheit und die – im Gegensatz zur festverdrahteten Technik – relativ geringe Betriebserfahrung sind einige Aspekte, welche bei der Bewertung der Zuverlässigkeit digitaler Einrichtungen betrachtet werden müssen.

Die nationalen und internationalen Regelwerke (u. a. KTA-Regeln, DIN-Normen, IAEA Safety Guides, IEC-Standards, U.S. NRC Regulatory Guides) spezifizieren bisher keine detaillierten, von allen Beteiligten anerkannten Nachweisverfahren und zugehörige Anforderungen zum Erreichen erforderlicher Zuverlässigkeit beim Einsatz digitaler Leittechnik in Kernkraftwerken. Infolgedessen wird auf diesem Gebiet seit mehr als 40 Jahren weltweit intensiv geforscht.

Im Rahmen des BMU-Vorhabens 3615R01343 („Entwicklung und Erprobung eines Werkzeugs zur Sensitivitätsanalyse der Fehlerauswirkungen in der sicherheitsrelevanten digitalen Leittechnik“) wurde daher eine modellbasierte Methodologie entwickelt und erprobt, mit der das dynamische Verhalten digitaler Leittechnik (DLT) beim Eintreten von systeminternen Fehlern analysiert werden kann. Auf Basis generischer Modelle moderner Systemarchitekturen wurde dabei durch Sensitivitätsanalysen der Einfluss unterschiedlicher Parameter (beispielsweise Reparaturzeiten, Redundanzgrad) auf die Zuverlässigkeit des Systems untersucht. /PIL 18/

[1] Die Sicherheitsanforderungen an Kernkraftwerke (SiAnf) in der Version vom März 2015 (/BMU 15/, /BMU 12a/) unterscheiden zwischen rechnerbasierten und programmierbaren leittechnischen Einrichtungen. Hierbei bestehen programmierbare Geräte definitionsgemäß aus mindestens einem diskreten programmierbaren Bauelement (die Anwendungsfunktion wird durch Verdrahtung oder durch Bauelementfunktionen realisiert), wogegen rechnerbasierte Geräte mindestens aus einem Prozessor bestehen (die Anwendungsfunktion ist im Speicher hinterlegt).

Im Rahmen des hier vorgestellten Vorhabens (4718R01314) wurde eine Reihe neuer Werkzeuge zur Untersuchung der Fehlerauswirkungen in sicherheitsrelevanter digitaler Leittechnik entwickelt, um die zuvor bereits vorhandene Methodologie nicht nur zu erweitern, sondern auch validieren zu können. Wesentlich hierfür war der Aufbau des Analyse- und Testsystems (AnTeS) der GRS. Dieses besteht aus unterschiedlichen Modulen und beinhaltet, neben simulierten Leittechniksystemen auch ein reales Leittechniksystem (basierend auf Hard- und Software des Teleperm-XS-Systems (TXS) der Firma Framatome) sowie einer Reihe von verfahrenstechnischen Simulationen. Das AnTeS wird im Kapitel 2 detailliert vorgestellt und ist eine äußerst flexible Testumgebung, die je nach Fragestellung, sehr unterschiedliche Analysen und Forschungsarbeiten ermöglicht. So wurde im Verlauf des Vorhabens deutlich, dass mit dem AnTeS auch in Zukunft Fragestellungen, beispielsweise zur generischen Verifizierung und Validierung digitaler Leittechnik, modellbasiert untersucht werden können.

Das Kapitel 3 befasst sich mit den neuen Werkzeugen und Methoden, die für und mit Hilfe des AnTeS im Rahmen des aktuellen Vorhabens entwickelt wurden. Deren Anwendung auf konkrete Fragestellungen ist schließlich das Thema des Kapitels 4, in welchem nicht nur die Validierung der Gesamtmethodologie, sondern die erweiterten Forschungsmöglichkeiten anhand der Beschreibung weiterer, zusätzlicher Tests dargestellt wird.

Im Kapitel 5 findet abschließend eine Zusammenfassung der durchgeführten Arbeiten und der dabei erlangten Ergebnisse statt und es wird ein Ausblick auf eine Reihe denkbarer zukünftiger Anwendungen gegeben.

2 AnTeS – das Analyse- und Testsystem der GRS

Im Rahmen des in diesem Dokument vorgestellten Vorhabens wurde das Analyse- und Testsystem der GRS weiterentwickelt. Das AnTeS besteht aus drei Modulen (vgl. Abb. 2.1) und kann flexibel für Forschungsarbeiten und Untersuchungen im Bereich der digitalen Leittechnik genutzt werden.

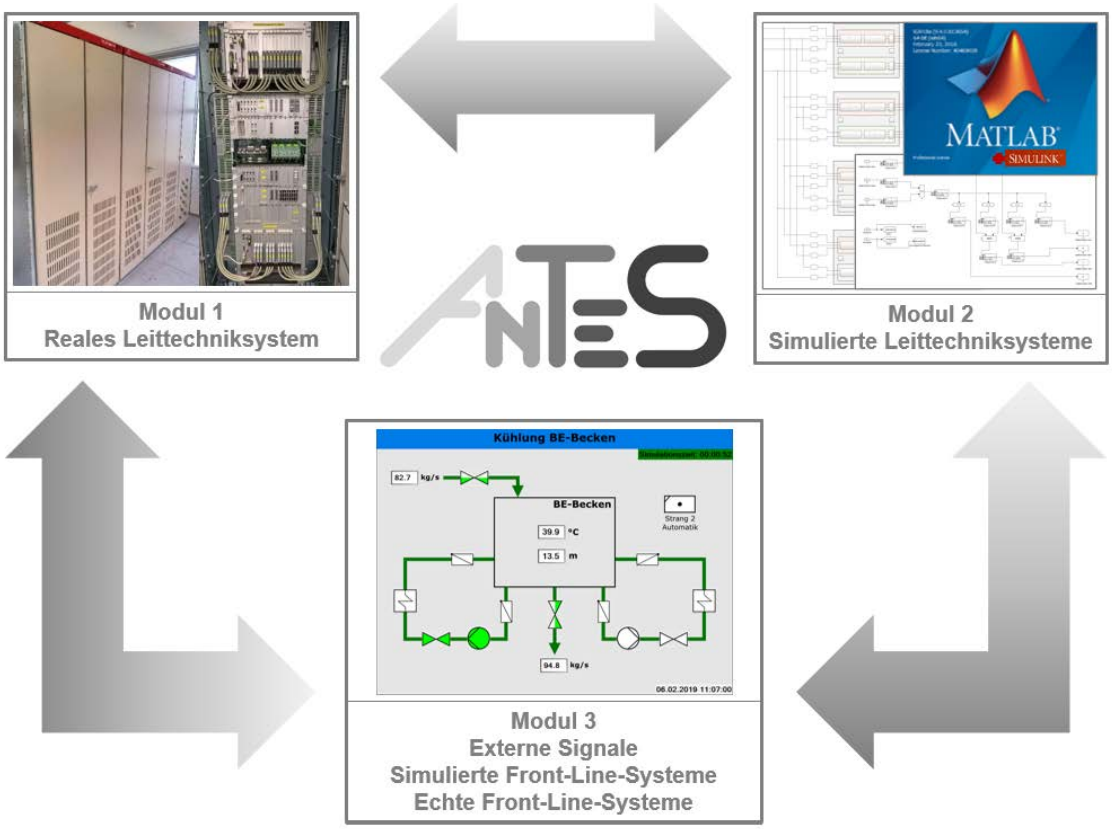


Abb. 2.1 Das Analyse- und Testsystem (AnTeS) der GRS.

Die grundlegenden Eigenschaften der einzelnen Module, die im Detail in den nachfolgenden Abschnitten erläutert werden, lassen sich wie folgt zusammenfassen.

- **Modul 1**: Reales Leittechniksystem
  - Basiert auf Komponenten des Teleperm XS
  - Hardware und Engineering-Software von Framatome (ehemals Areva)
  - Unterschiedliche Leittechnikarchitekturen realisierbar
  - Unterschiedliche Netzwerktopologien realisierbar
  - Typische leittechnische Funktionen der Prozessautomatisierung und für Sicherheitsaufgaben umsetzbar
  - Beeinflussung von Signalen und Eingabe von Fehlern möglich (Fault-Injection)

  - Automatisierte Auswirkungsanalysen von Fehlerkombinationen (z. B. zur V&V (Verification and Validation) einer Failure Mode and Effects Analysis) für alle Eingangssignalkombinationen und zuvor festgelegte Ausfallarten innerhalb des Leittechniksystems möglich
  - Online-Überwachung sämtlicher Signale innerhalb der Leittechnik möglich

- **Modul 2**: Simulierte Leittechniksysteme
  - Abbildung generischer oder der mit realen Leittechniksystemen umgesetzten Funktionen mit der Software Matlab/Simulink /MAT 21/
  - Beeinflussung von Signalen und Eingabe von Fehlern möglich (Fault-Injection)
  - Automatisierte Auswirkungsanalysen von Fehlerkombinationen (z. B. zur V&V einer Failure Mode and Effects Analysis) für alle Eingangssignalkombinationen und zuvor festgelegte Ausfallarten innerhalb des Leittechniksystems möglich
  - Durchführung von Monte-Carlo-Simulationen (statistische Fehlereinspeisung und Ergebnisaufzeichnung für lange simulierte Zeiträume) möglich

- **Modul 3**: Generierung und Überwachung von Ein- und Ausgangssignalen
  - Für die Module 1 und 2 über zusätzliche Hardware
  - Software entwickelt in der Programmiersprache Python (flexibel änderbar)
  - Manuelle oder automatische (per Skript-Steuerung) Vorgabe von Eingangssignalen und Aufzeichnung der durch die Leittechnik-Systeme (Modul 1 und 2) generierten Ausgangssignale
  - Anbindung von Simulationen verfahrenstechnischer Systeme, z. B.
    - Simulation eines generischen Brennelementlagerbeckens
    - Simulation eines generischen Pool-Reaktors
    - Simulation eines Maschinenhauskrans
  - Anbindung realer externer Systeme

2.1 Modul 1: Reales Leittechniksystem

Das Modul 1 des AnTeS basiert auf Komponenten des Teleperm-XS-Systems der Firma Framatome (zuvor: Areva), welche im Mai 2017 von der GRS beim Kernkraftwerk Krümmel (KKK) erworben und zum GRS-Standort Garching geliefert wurden (siehe Abb. 2.2). Ursprünglich waren diese Komponenten für eine Modernisierung der Turbinensteuerung im KKK vorgesehen, kamen dort aber nicht mehr zum Einsatz. Aufgrund der beschlossenen endgültigen Stilllegung des Kraftwerks im Jahr 2011 wurden diese vom KKK nicht mehr benötigt und daher an die GRS veräußert.

Abb. 2.2 Anlieferung des Teleperm-XS-Systems ab Mai 2017 bei der GRS in Garching (linkes Bild).

Drei der Leittechnikschränke wurden in einem Serverraum der GRS installiert (rechtes Bild), der vierte Schrank enthält das Servicegerät und wurde in einem separaten Raum aufgebaut.

Nach dem Aufstellen der drei TXS-Leittechnikschränke in einem Serverraum der GRS in Garching und dem Anschließen einer geeigneten Stromversorgung (3-fache Gleichstromversorgung mit 24 V), wurden zwei der Schränke vollständig entkernt und anschließend das Modul 1 des AnTeS in diesen Schränken aufgebaut, in Betrieb genommen und auf Funktionsfähigkeit überprüft.

Abb. 2.3 Modul 1 des AnTeS

Zwei der bei der GRS vorhandenen TXS-Schränke nach dem vollständigen Neuaufbau.

Die in diesem Vorhaben zumeist verwendete Konfiguration des Moduls 1 des AnTeS umfasst bis zu vier redundante Verarbeitungseinheiten (bestehend aus jeweils einer Prozessorbaugruppe sowie analogen und digitalen Ein- und Ausgabebaugruppen) in zwei Schränken (vgl. Abb. 2.3). Jeder Leittechnikschrank verfügt darüber hinaus über gesonderte Kommunikationsprozessorbaugruppen, über die eine Netzwerkverbindung mit dem sogenannten Servicerechner hergestellt werden kann. Abb. 2.4 zeigt eine mögliche Konfiguration (Netzwerkplan) in der Engineering-Umgebung SPACE (Specification and Coding Environment) des TXS. SPACE selbst erlaubt die flexible Erstellung unterschiedlicher Konfigurationen und die Programmierung des TXS (Erstellung von Funktionsplänen) auf dem Servicerechner über eine grafische Oberfläche (vgl. Abb. 2.5). Hierfür wurde der ebenfalls aus dem KKK mitgelieferte Servicerechner virtualisiert, dieser wird nun als virtuelle Maschine auf einem Standard-PC in einem separaten Schrank innerhalb desselben Serverraums betrieben.

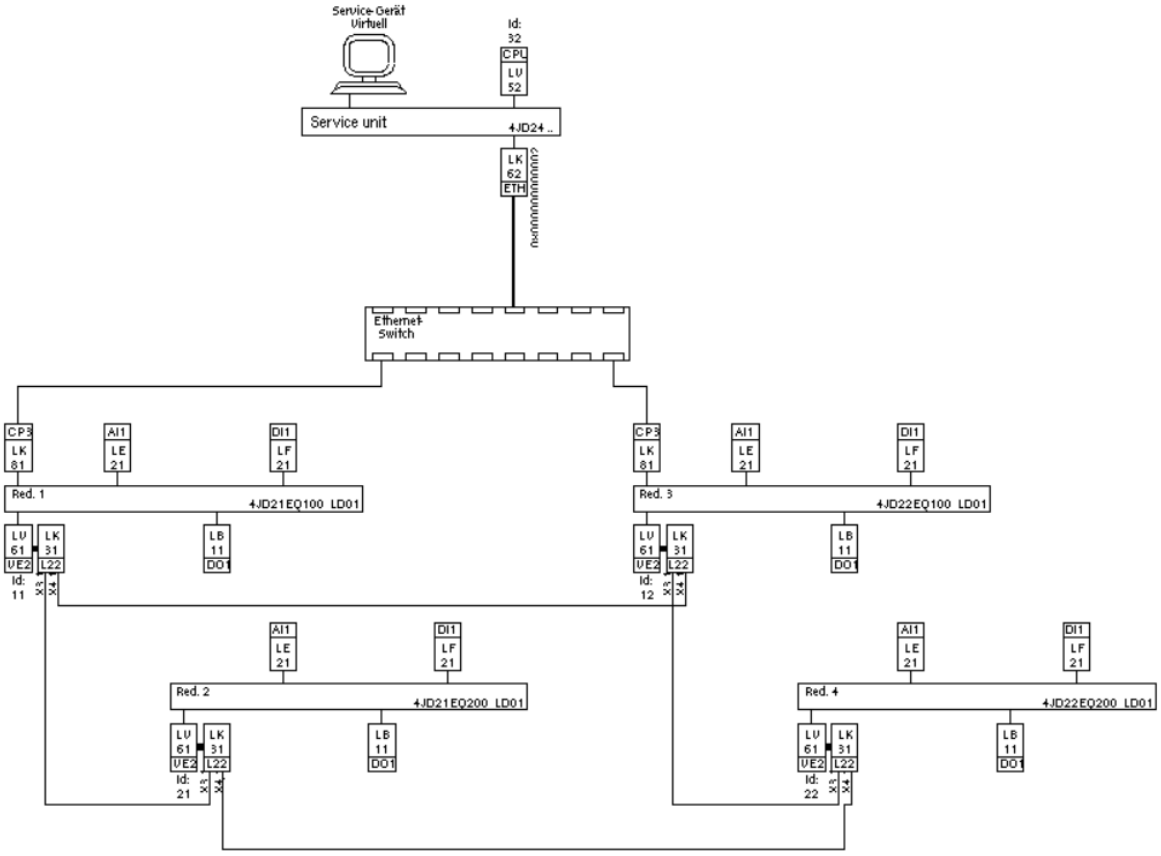


Abb. 2.4 Eine mögliche Konfiguration (Netzwerkplan) von Modul 1 des AnTeS

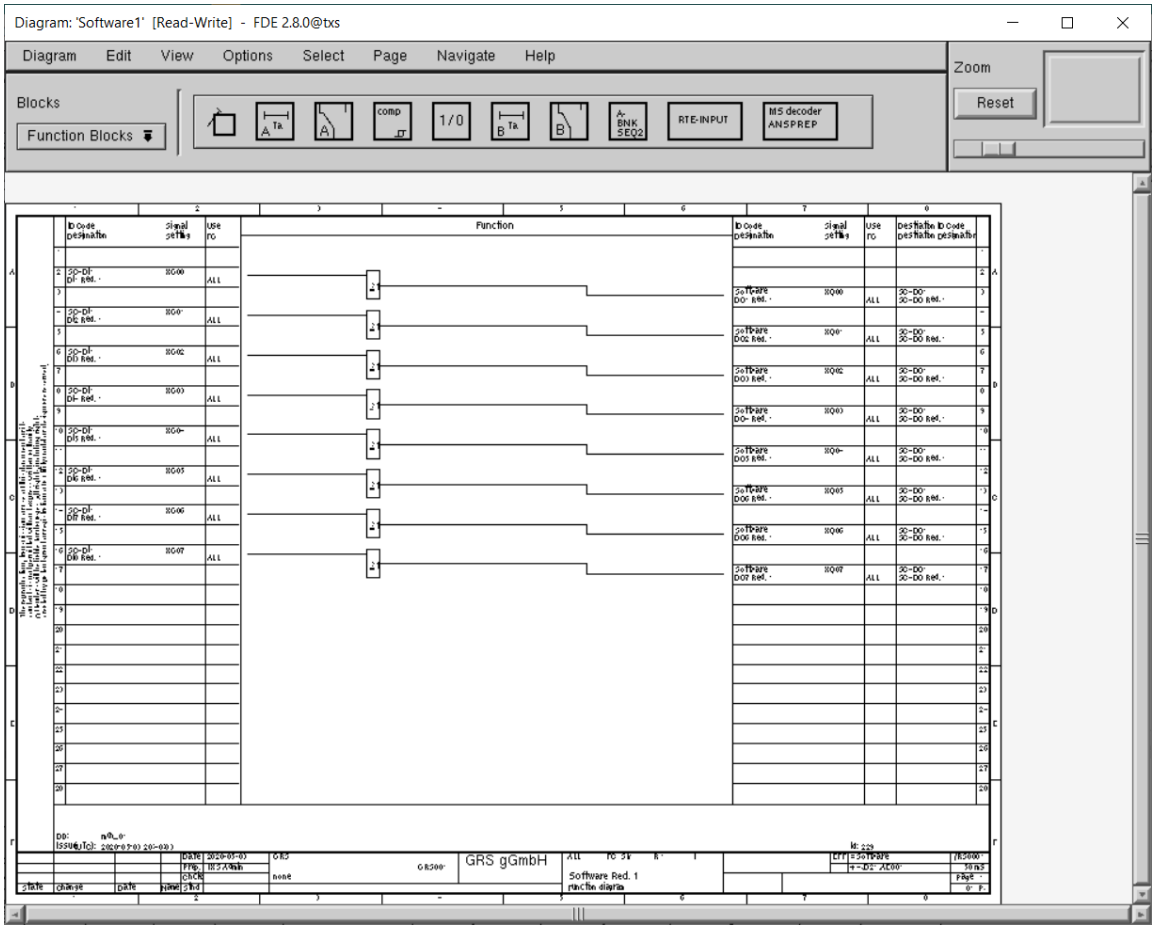


Abb. 2.5 Typische Oberfläche von SPACE (Specification And Coding Environment) zum Konfigurieren und Programmieren des TXS auf dem Servicerechner.

2.2 Modul 2: Simulierte Leittechniksysteme

Neben der Verwendung eines realen Leittechniksystems (Modul 1), erlaubt das Modul 2 des AnTeS die Verwendung simulierter Leittechniksysteme für Analysen. Dabei können sowohl dieselben Funktionalitäten wie im realen Leittechniksystem (TXS) als auch andere reale sowie generische Leittechniksysteme simuliert werden.

Das Modul 2 des AnTeS basiert auf der Software Simulink, diese kann allgemein zur Modellierung von Systemen genutzt werden und ist ein Zusatzprodukt zur Software Matlab /MAT 21/. Die Modellierung der Systeme erfolgt mit Hilfe von grafischen Funktionsblöcken. Der Datenfluss zwischen diesen Funktionsblöcken wird über Verbindungslinien realisiert. Ein auf diese Weise modelliertes System kann anschließend innerhalb von Simulink mit Hilfe unterschiedlicher Lösungsverfahren („Solver“) simuliert werden.

Zusätzliche Softwarepakete von Simulink erlauben es, diese Simulationen als sogenannte dynamische Programmbibliotheken (Dynamic Link Library – DLL) zu exportieren, welche dann z. B. durch externe eigene Programme oder Skripte für die weitere Simulation verwendet werden können (Für Details hierzu siehe Anhang A.2). Alternativ erlaubt Simulink auch den direkten Export und Betrieb von Simulationsmodellen auf externer Hardware (beispielsweise auf Raspberry-Pi-Mikrocomputern), wodurch z. B. die Verwendung der Modelle mit weiteren Systemen (z. B. innerhalb des Moduls 3 von AnTeS) vereinfacht wird.

Die Verwendung von Simulink zur Programmierung von Leittechnik-Funktionen ähnelt in der Handhabung der Engineering-Umgebung SPACE des Teleperm-XS-Systems. Innerhalb der Softwareoberfläche stehen, abhängig von den vorhandenen Lizenzen, über den Simulink Library Browser eine Vielzahl von Funktionsblöcken zur Verfügung (vgl. Abb. 2.6).

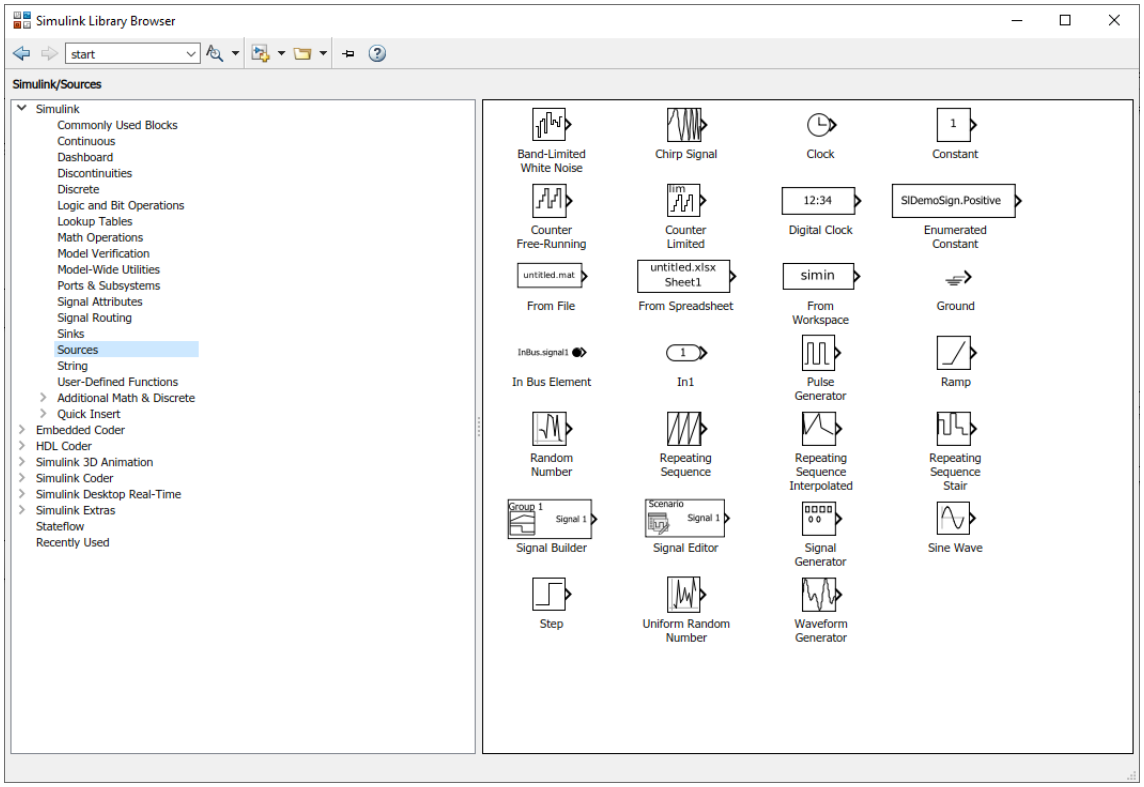


Abb. 2.6 Der Simulink Library Browser zur Auswahl von Funktionsblöcken

Mit Hilfe der verfügbaren Funktionsblöcke können sehr unterschiedliche Modellsysteme realisiert werden (auch gänzlich andere Systeme als Leittechniksysteme).

Für die folgenden Ausführungen ist von besonderer Bedeutung, dass über sogenannte Subsysteme auch eigene Funktionsblöcke erstellt werden können. Für die Verwendung von Simulink als Engineering-Umgebung virtueller (simulierter) Leittechniksysteme, können auf diese Weise also auch solche Funktionsblöcke erstellt werden, die sich in der gleichen Weise verhalten wie die entsprechenden Funktionsblöcke realer Leittechniksysteme. Für die Nachbildung (Erstellung) von TXS-Funktionsplänen mit Hilfe von Simulink, wurde daher eine Vielzahl von TXS-Funktionsblöcken (als Simulink-Subsysteme) erstellt. Auf diese Weise entstand eine eigene Simulink-Bibliothek mit TXS-Funktionsblöcken, die bereits eine Vielzahl (der am häufigsten verwendeten) Bausteine des TXS enthält, aber noch keineswegs vollständig ist. Fehlende Bausteine können jedoch bei Bedarf jederzeit mit überschaubarem Aufwand ergänzt werden, sobald diese für eine Anwendung benötigt werden. Darüber hinaus können in Simulink aber auch Funktionalitäten der Hardware der simulierten Leittechniksysteme bei Bedarf über vorhandene Funktionsblöcke von Simulink oder selbst erstellte Subsysteme berücksichtigt werden.

Nachfolgend wird repräsentativ anhand des Beispiels eines sogenannten „2. MAX“-Funktionsblocks („zweites Maximum“) verdeutlicht, wie Funktionsblöcke in Anlehnung ans TXS in Simulink erstellt wurden. Derselbe Funktionsblock wie im TXS steht damit auch für die Erstellung von Funktionsplänen simulierter Leittechniksysteme zur Verfügung (vgl. Abb. 2.7).

Abb. 2.7 Beispiel für Nachbildung eines TXS-Funktionsblocks im Modul 2 des AnTeS (Matlab/Simulink)

In diesem Beispiel des Bausteins „2. MAX“

Im Handbuch des TXS /TXS 12/ wird das Verhalten des Funktionsblocks „2. MAX“ wie folgt beschrieben (ins Deutsche übersetzt, Auszug):

Der Funktionsblock 2.MAX wählt das zweite Maximum von bis zu vier analogen Eingangssignalwerten aus. Das 2. Maximum kann berechnet werden, indem die Werte in aufsteigender Reihenfolge sortiert und dann der zweite Wert von oben ausgewählt wird (das 2. Maximum von {4, 3, 2, 1} ist 3, das 2. Maximum von {4, 4, 2, 1} ist 4).

Der Funktionsbaustein passt seinen Betriebsmodus and die Anzahl der Eingangssignale an, für die das ERROR-Attribut gesetzt ist:

???? *Wenn für kein Eingangssignal das ERROR-Attribut gesetzt ist, wird das 2. Maximum ausgewählt.*

???x *Wenn für ein Eingangssignal das ERROR-Attribut gesetzt ist, wird das 2. Maximum der verbleibenden Eingangssignale ausgewählt.*

??xx *Wenn für zwei Eingangskanäle das ERROR-Attribut gesetzt ist, dann wird das 2. Maximum der verbleibenden Eingangssignale ausgewählt, d. h. das Minimum der verbleibenden zwei.*

?xxx *Wenn für drei Eingangssignale das ERROR-Attribut gesetzt ist, wird das verbleibende Eingangssignal weitergeleitet.*

xxxx *Wenn für alle vier Eingangssignale das ERROR-Attribut gesetzt ist, wird das ERROR-Attribut des Ausgangssignals AO1 gesetzt und dem Ausgangswert von AO1 der Wert 0,0 zugewiesen.*

Das ERROR-Attribut ist eine zusätzliche Information, die an ein Signal innerhalb des TXS angehängt wird, um dessen Validität bzw. Invalidität (z. B. bei erkannten Fehlern) zu kennzeichnen. Jedes „analoge“ bzw. „digitale“ Signal setzt sich aus zwei Informationen zusammen: Zum einen aus dem eigentlichen Signalwert, zum anderen aus einem binären Wert (0/1), der die Validität des Signals kennzeichnet. Der entsprechende Signalanteil wird häufig, wie z. B. auch im Vorgängervorhaben, auch als Error-Flag bezeichnet. Ist das Error-Flag gesetzt (1), so ist nach der Sprechweise der Funktionsblock-Beschreibung oben das ERROR-Attribut gesetzt.

Anmerkung: Bei der Beschreibung des Funktionsblocks „2. MAX“ (s. o.) werden die Begriffe „analog“ und „digital“ unpräzise verwendet, was in der Leittechnik-Fachwelt häufig vorkommt. Selbstverständlich sind alle Signale innerhalb des eigentlichen Leittechniksystems (TXS oder auch in Simulink) immer digital (und niemals analog).

Häufig werden die Begriffe „analog“ und „digital“ dazu verwendet, um zwischen rein binären Signalen (z. B. 0/1; 0 V/24 V) und solchen Signalen zu unterscheiden, die mehr als nur zwei Zustände kennen (z. B. Gleitkommazahlen, 4...20 mA, ...). Aus diesem Grund wurden die Begriffe oben in Anführungszeichen gesetzt.

Die Beschreibung des Verhaltens der „2. MAX“-Funktionsblöcke wurde in Simulink, wie bereits erwähnt, durch ein Subsystem realisiert. Eine Übersicht der Logik des Subsystems ist in Abb. 2.8 dargestellt.

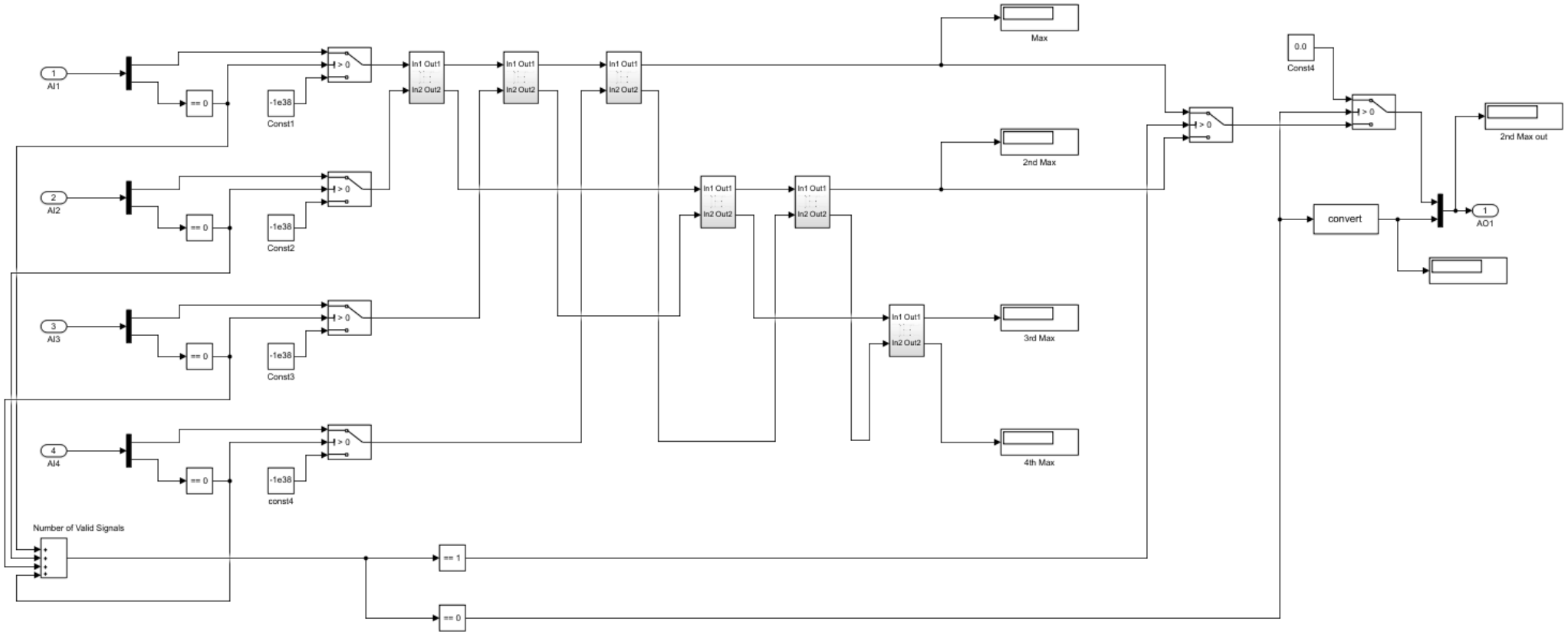


Abb. 2.8 Interne Logik (Simulink-Subsystem) des Funktionsblocks „2. MAX“ in Abb. 2.7.

Auf der linken Seite des entsprechenden Subsystems sind die vier Eingänge (AI1, AI2, AI3, AI4) zu sehen. Die Bezeichnung dieser Eingänge folgt der im TXS üblichen Nomenklatur, hierbei steht AI für „Analog In“. Auf der rechten Seite befindet sich der Ausgang AO1 (AO für „Analog Out“).

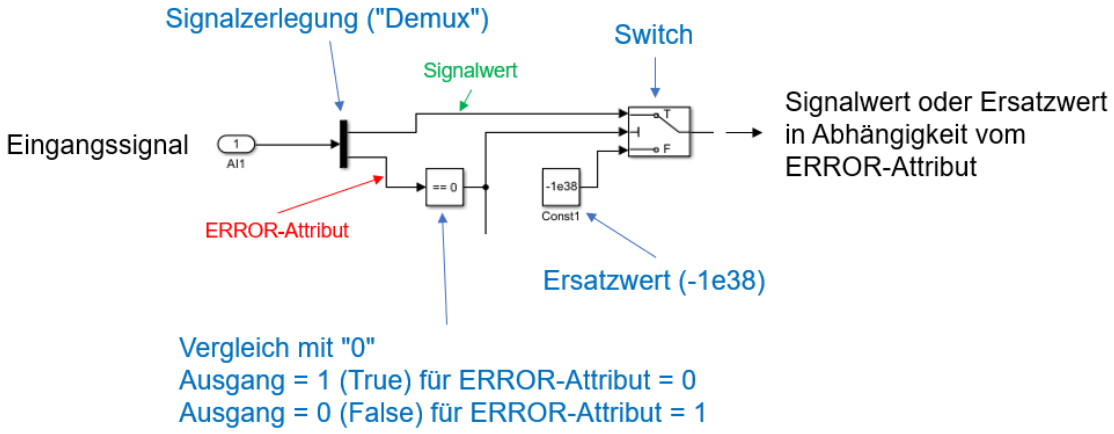


Abb. 2.9 Detailansicht der internen Logik des Bausteins „2. MAX“.

Abb. 2.9 zeigt eine Detailansicht der internen Logik des Funktionsblocks „2. MAX“ aus Abb. 2.8. Hier lässt sich am Beispiel des Eingangssignals AI1 erkennen, wie die Eingangssignale vorverarbeitet werden, bevor diese anschließend zur Bildung des 2. Maximums sortiert werden (siehe weiter unten). Zunächst werden, wie in dieser Detailansicht für AI1 zu erkennen, die Signale in ihre zwei Bestandteile zerlegt (Signalwert und Validität bzw. ERROR-Attribut). In Abhängigkeit vom ERROR-Attribut (0 oder 1) werden durch Schalter („Switches“) jeweils entweder der tatsächlichen Eingangswert oder eine Konstante (Wert: $-1 \cdot 10^{38}$) für die Bestimmung des 2. Maximums weitergeleitet.

Anschließend werden die vier vorverarbeiteten Signale der Größe nach sortiert. Hierzu werden die Werte entsprechend eines Sortieralgorithmus (siehe z. B. /WIK 21/) nacheinander paarweise verglichen und ggfs. vertauscht. Beispielhaft wird dies Abb. 2.10 verdeutlicht. Darin wird die Zahlenfolge {1,8,9,5} durch eine Abfolge von paarweisen Vergleichen von Zahlenwerten in die korrekte (abfallende) Reihenfolge {9,8,5,1} sortiert. Der jeweilige Vergleich und ggfs. Tausch zweier Zahlenwerte wird dabei jeweils durch ein weiteres Simulink-Subsystem durchgeführt, dessen Verhalten ebenfalls beispielhaft in Abb. 2.11 detaillierter dargestellt wird.

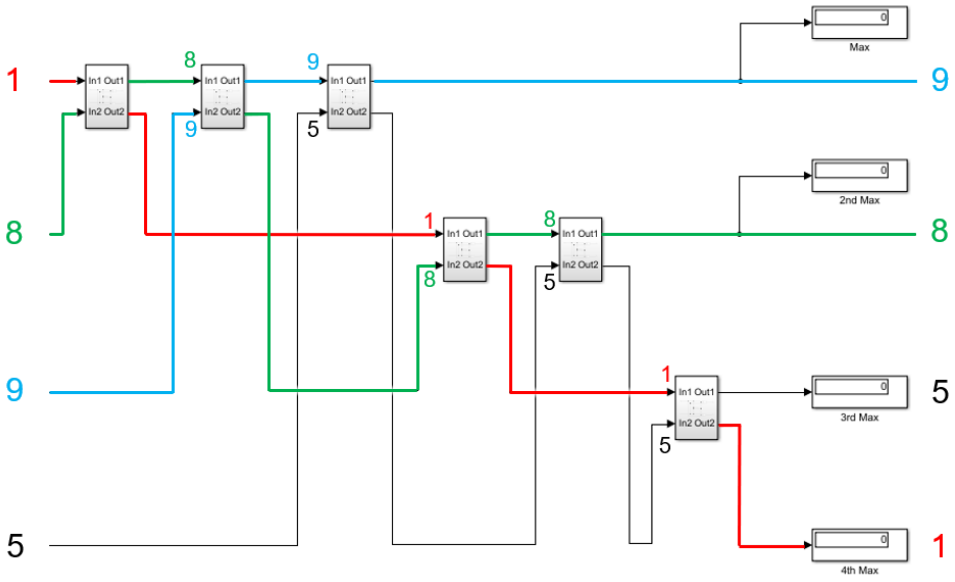


Abb. 2.10 Beispiel für die Sortierung der vier Eingangswerte im Funktionsblock „2.MAX“ (Detailansicht aus Abb. 2.8).

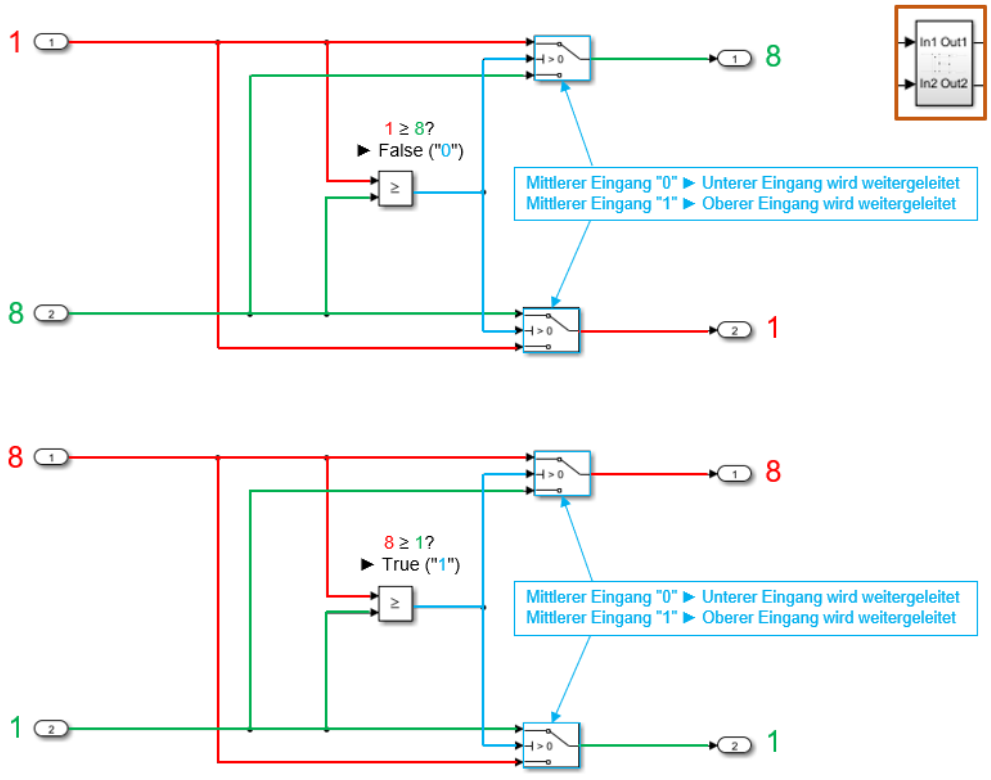


Abb. 2.11 Zwei repräsentative Beispiele für den Vergleich von {1, 8} bzw. {8, 1} mit einem Subsystem innerhalb der Logik des „2. MAX“-Bausteins.

In Abhängigkeit von der Anzahl der Eingangssignale und davon, für wie viele der Eingangssignale das ERROR-Attribut gesetzt ist, soll der erstellte Funktionsblock (wie im Vorbild TXS) das zweite Maximum, den maximalen Wert oder eine Konstante („0.0“) ausgeben und entsprechend der Beschreibung des Funktionsblocks auch ggfs. das ERROR-Attribut des Ausgangssignals gesetzt werden. Hierzu dienen bisher noch nicht beschriebenen Teile der Logik in Abb. 2.8. Zu diesem Zweck wird mit Hilfe eines Additionsbausteins von Simulink die Anzahl der validen Signale bestimmt (Abb. 2.12). Auf Basis der so bestimmten Anzahl valider Signale wird dann, wie in Abb. 2.13 dargestellt, der Ausgabewert und die Validität des Ausgangs AO1 bestimmt.

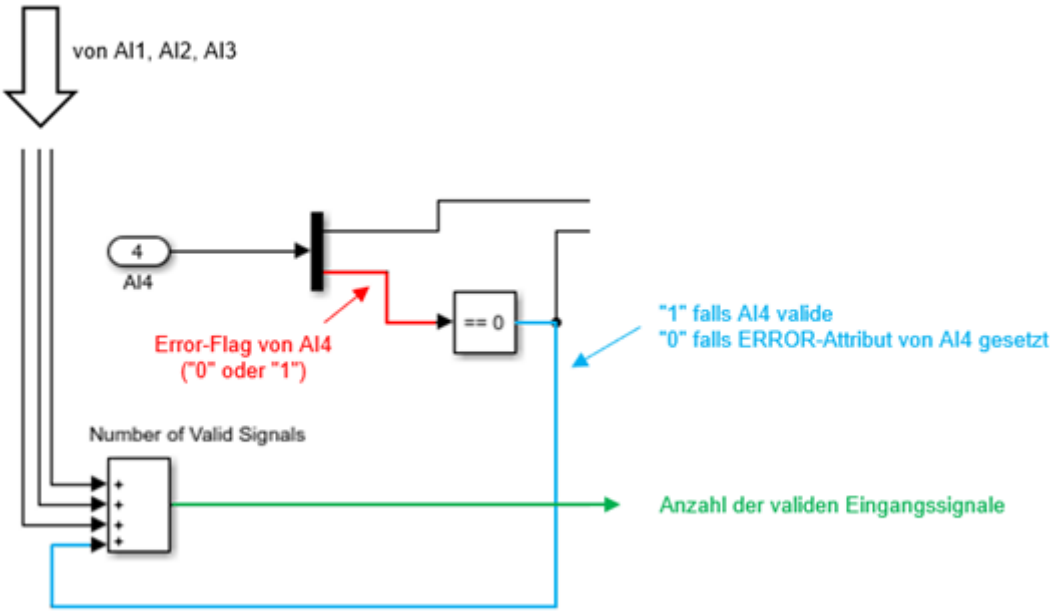


Abb. 2.12 Bestimmung der Anzahl der validen Eingangssignale im Funktionsbaustein „2. MAX“.

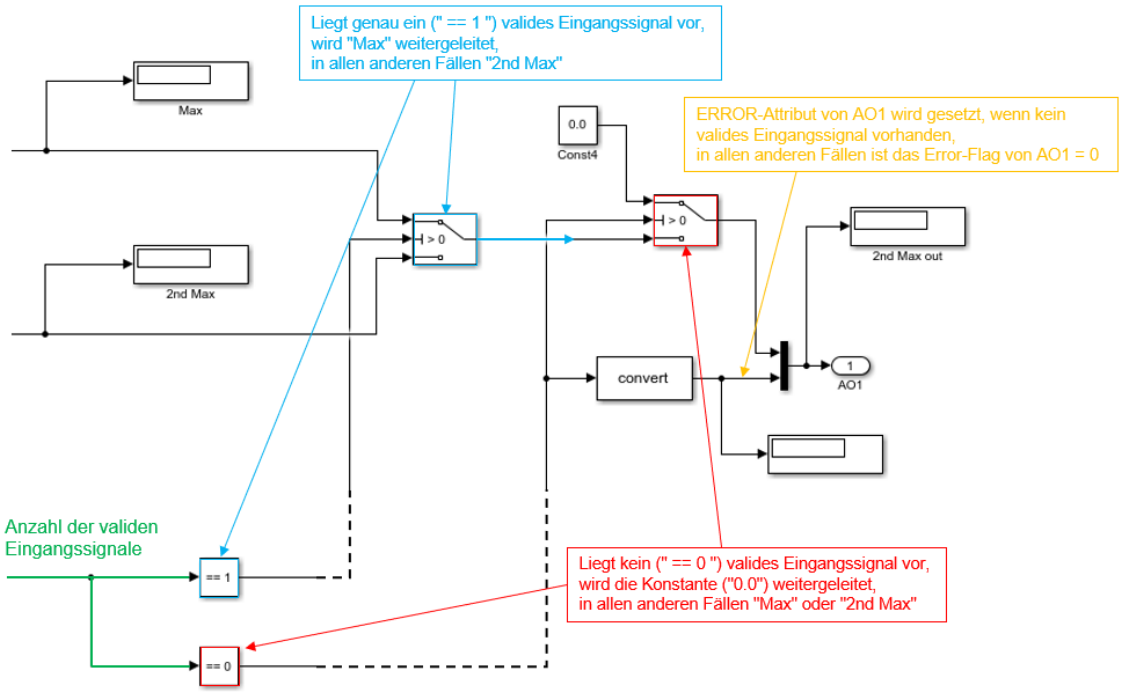


Abb. 2.13 Bestimmung des Ausgabewertes von AO1 („Max“, „2nd Max“ oder „0.0“) und dessen Error-Flags („0“ oder „1“) in Abhängigkeit von der Anzahl der validen Eingangssignale in Abb. 2.8.

In derselben Weise wurden alle für dieses Vorhaben wesentlichen Funktionsblöcke des TXS im Modul 2 des AnTeS umgesetzt. Bei Bedarf können weitere Bausteine (z. B. generischer Leittechniksysteme) innerhalb kurzer Zeit realisiert werden.

Simulationen leittechnischer Systeme innerhalb des Moduls 2 von AnTeS können nach Erstellung der entsprechenden leittechnischen Funktionen (und ggfs. Berücksichtigung des Einflusses relevanter Hardware) anschließend direkt aus der Software Simulink heraus entweder auf eigens hierfür vorgesehener Hardware (z. B. Raspberry Pis) übertragen oder in sogenannte Direct Link Libraries (DLLs) für die weitere Verwendung umgewandelt werden. Die Erstellung von DLLs wird im Anhang A.2 erläutert, diese können z. B. für die Durchführung automatisierter Auswirkungsanalysen von Fehlerkombinationen (siehe Abschnitt 3.2) oder für Monte-Carlo-Simulationen (siehe Abschnitt 3.3) verwendet werden.

2.3 Modul 3: Anbindung externer Signale

Das Modul 3 dient der Ein- und Ausgabe von Signalen in die und aus den Modulen 1 und 2 des AnTeS. Es übernimmt dabei neben der Anzeige und Bedienung der Leittechniksysteme auch alle weiteren Aufgaben, die nicht unmittelbar den Modulen 1 und 2 zugeordnet werden können (beispielsweise verfahrenstechnische Simulationen oder die Anbindung realer verfahrenstechnischer Systeme). Wesentlich hierfür war die Entwicklung spezieller Schnittstellen (Interfaces), die im nachfolgenden Abschnitt 2.3.1 näher beleuchtet werden. Sowohl in den Mikrorechnern, die in diese Interfaces integriert sind, als auch allgemein auf PCs, können darüber hinaus verfahrenstechnische Simulationen ausgeführt und mit den Leittechniksystemen der Module 1 und 2 verbunden werden (vgl. Abschnitte 2.3.2 und 2.3.3). Zusätzlich stehen aber auch einige reale Systemkomponenten bei der GRS zur Verfügung, die ebenfalls mit den Leittechniksystemen der Module 1 und 2 gekoppelt werden können (vgl. Abschnitt 2.3.4).

2.3.1 AnTeS-Interfaces

Ein Leittechniksystem wird in praxisbezogenen Anwendungen nicht für sich allein betrieben. Vielmehr dient es typischerweise der Steuerung anderer Systeme und muss dafür mit diesen kommunizieren können (z. B. also gemessene Prozessparameter anzeigen und verarbeiten, Steuerbefehle vom Bediener entgegennehmen und Befehle an Aktuatoren schicken). Im Modul 1 des AnTeS werden hierzu Ein- und Ausgangskarten des TXS mit den folgenden Eigenschaften verwendet:

- **SDI**
  - TXS Digital Input Module
  - 32 binäre Eingangskanäle (+24 V)
- **SDO**
  - TXS Digital Output Module
  - 32 binäre Ausgangskanäle (+24 V)

- **SAI**
  - TXS Analog Input Module
  - 8 analoge Eingangskanäle (bei Differenzmessung zwischen jeweils zwei Eingängen, konfigurierbar für Spannungssignale bis ±10 V oder Stromsignale bis ±20 mA bzw. +50 mA)
  - 16 analoge Eingänge (bei Messung gegen Erdpotential, konfigurierbar für Spannungssignale bis ±10 V oder Stromsignale bis ±20 mA bzw. +50 mA)
- **SAO**
  - TXS Analog Output Module
  - 8 analoge Ausgangskanäle (konfigurierbare Ausgabe von Spannungssignalen (bis ±10 V) oder Stromsignalen (bis +50 mA))

Um gezielt Eingangssignale für das Modul 1 des AnTeS zu generieren, aber auch für das Auslesen der vom Leittechniksystem ausgegebenen Signale, wurden zwei AnTeS-Interfaces aufgebaut. Diese wurden derart konstruiert, dass auch verfahrenstechnische Simulationen unkompliziert und rückwirkungsfrei an das Modul 1 des AnTeS angebunden bzw. wahlweise auch direkt auf Interface-internen Rechnern betrieben werden können. Beide AnTeS-Interfaces sind prinzipiell gleich aufgebaut und befinden sich als eigene Einschübe in den Racks der Leittechnikschränke des Moduls 1 des AnTeS (vgl. Abb. 2.14).

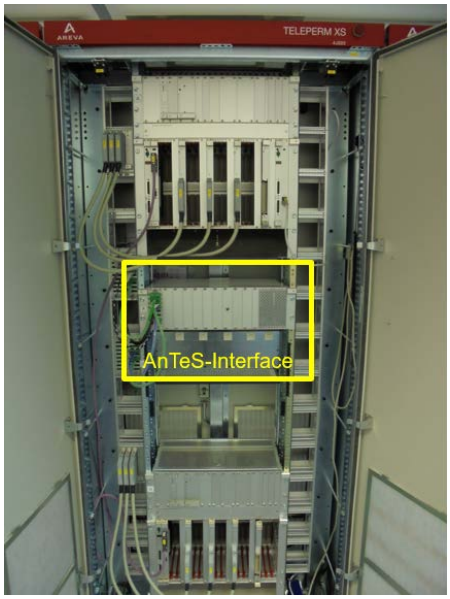


Abb. 2.14 Eines der beiden AnTeS-Interfaces in einem der beiden TXS-Schränke

Abb. 2.15 zeigt den Aufbau eines der beiden Interfaces im geöffneten Zustand von vorne und hinten. In den nummerierten Bereichen sind im Einzelnen die folgenden Komponenten verbaut:

1. SCM2-Baugruppe (TXS-Baugruppe)
    - Es handelt sich um einen einfachen Einschub zur Signalrangierung ohne aktive Bauteile, dieser besteht nur aus Buchsen für den Anschluss von standardisierten Steckern.
2. Zwei Optokopplerkarten mit jeweils acht Kanälen
    - Diese dienen hauptsächlich der Umwandlung der vom TXS ausgegebenen binären 24-V-Signale in 3,3-V-Signale (stellen aber auch gleichzeitig eine elektrische Entkopplung zwischen Interface und TXS dar).
3. Zwei Relaiskarten mit jeweils acht Kanälen
    - Diese wandeln die im Interface generierten 3,3-V-Binärsignale in 24-V-Signale für das TXS um (und stellen ebenfalls eine Entkopplung zwischen Interface und TXS dar).
4. Mikrocontroller-Karte
    - Typ: Arduino Due
    - 54 digitale (binäre) Ein- und Ausgänge (3,3 V, zwischen Input und Output umschaltbar), davon werden im Interface 16 als binäre Eingänge und 16 als binäre Ausgänge verwendet.
    - 12 analoge Eingangskanäle (mit A/D-Umwandlung), davon sind 8 für zukünftige Anwendungen im Interface verdrahtet worden (werden aber derzeit noch nicht genutzt).
    - 2 analoge Ausgangskanäle (mit D/A-Umwandlung, zwischen 0,56 V und 2,76 V frei einstellbar), diese dienen der Generierung von analogen Signalen für das Leittechniksystem.
5. Mikrorechner
    - Typ: Raspberry Pi 3 B+
    - Seriell mit dem Arduino Due zu dessen Steuerung verbunden.
    - Auf diesem läuft von der GRS entwickelte Software (Python). Über diese Software können die Eingangssignale ins TXS erzeugt und ausgegeben sowie Ausgangssignale aus dem TXS ausgelesen werden.
    - Auf diesem können auch unmittelbar verfahrenstechnische Simulationen ausgeführt werde.

- o Durch Anbindung an ein eigenes Netzwerk können auch Simulationen auf anderen Computern oder beliebige andere Signalquellen und -ziele mit diesem Mikrorechner kommunizieren. Hierüber ist auch die Anbindung simulierter Leittechniksysteme (Modul 2 des AnTeS) möglich.

6. Zwei interne Netzteile (+5 V und +24 V)

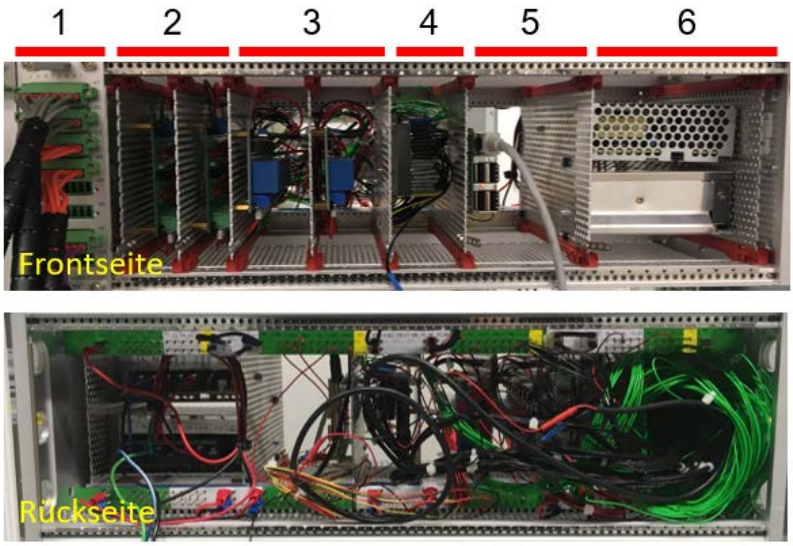


Abb. 2.15 Front- und Rückseite eines AnTeS-Interfaces im geöffneten Zustand

Die Verschaltung der einzelnen Komponenten des AnTeS-Interfaces ist in Abb. 2.16 dargestellt. Die Optokopplerkarten nehmen binäre Signale aus dem Leittechniksystem aus der Ausgangskarte (SDO) entgegen (0…3 V für eine logische 0 (False), 24 V für eine logische 1 (True)). Durch die an den Optokopplerkarten anliegende Versorgungsspannung von 3,3 V werden diese auf das Spannungsniveau der Mikrocontroller-Karte umgewandelt (0 V für eine logische 0 (False), 3,3 V für eine logische 1 (True)). Die insgesamt 16 Ausgänge der beiden Optokopplerkarten sind dementsprechend mit 16 binären Eingängen der Mikrocontroller-Karte (Typ: Arduino Due) verbunden. Umgekehrt verbinden die beiden Relaiskarten 16 binäre Ausgänge der Mikrocontroller-Karte (0 V, 3,3 V) mit 16 Eingängen des Leittechniksystems (SDI). Die Mikrocontroller-Karte verfügt außerdem über zwei analoge Ausgänge mit D/A-Wandlern, mit diesen können direkt analoge Eingangssignale für das Leittechniksystem generiert werden (SAI).

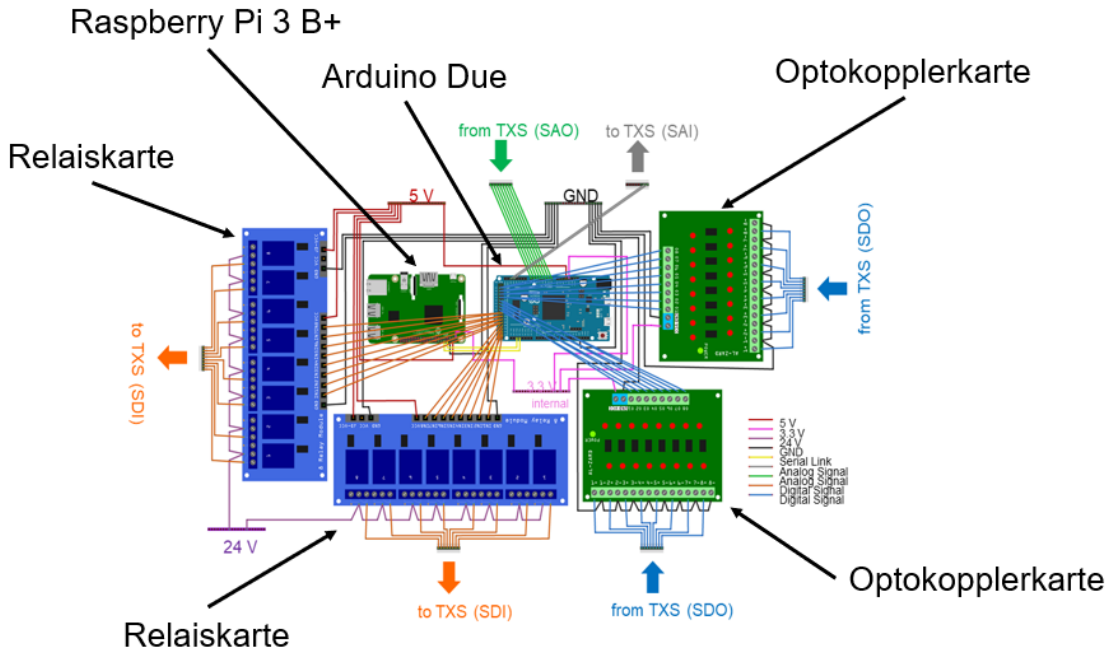


Abb. 2.16 Schaltplan der AnTeS-Interfaces

Für eine flexible und unkomplizierte Steuerung wird die Mikrocontroller-Karte über einen eigens hierfür verbauten Mikrorechner (Typ: Raspberry Pi 3 B+) gesteuert, dieser ist seriell mit der Mikrocontroller-Karte verbunden. Die entsprechende Software ist bei der GRS mit der Programmiersprache Python entwickelt worden und erlaubt somit auch zukünftig flexible Änderungen. Die Mikrorechner der Interfaces sind über eine Ethernet-Verbindung auch ans eigene AnTeS-Netzwerk angeschlossen, wodurch diese über Remoteverbindungen gesteuert, aber auch mit anderen Signalquellen verbunden werden können (u. a. Simulationen, aber auch reale verfahrenstechnische Komponenten (siehe Abschnitt 2.3.4)).

In der einfachsten Anwendung können also in den Interfaces über die entsprechende Software jeweils (bis zu) 16 binäre Eingangssignale und zwei analoge Eingangssignale für das Leittechniksystem erzeugt und gleichzeitig bis zu 16 binäre Ausgangssignale des Leittechniksystems eingelesen werden. Die Nutzeroberfläche der Software ist in Abb. 2.17 dargestellt. Diese erlaubt zusätzlich für gezielte Analysen auch die Verwendung von Skripten zur Ansteuerung und zum Speichern von Ergebnissen.

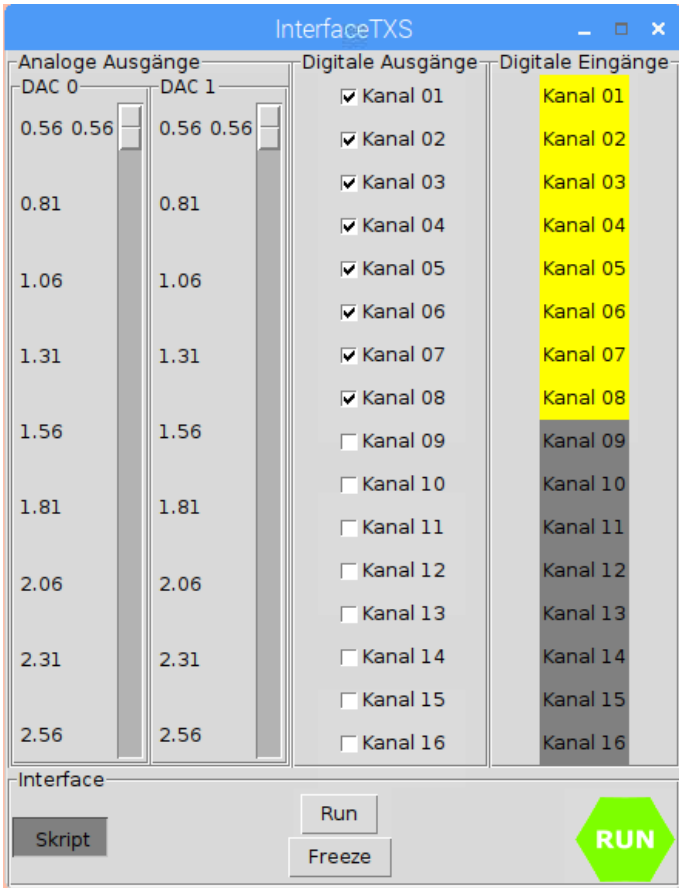


Abb. 2.17 Software der Interfaces zum Generieren von Eingangssignalen und zum Auslesen von Ausgangssignalen des Leittechniksystems (Modul 1 des AnTeS).

Insgesamt stehen also zwei von der GRS entwickelte, flexible Interfaces im Modul 3 des AnTeS zur Verfügung, mit denen zusammen insgesamt u. a. die folgenden Funktionen durchführbar sind:

- Generierung von bis zu 32 frei einstellbaren oder per Skript-Steuerung automatisch steuerbaren binären Signalen als Eingangssignale in das Modul 1 des AnTeS (TXS).
- Einlesen (und bei Skript-Steuerung aufzeichnen) von bis zu 32 binären Ausgangssignalen aus dem Modul 1 des AnTeS (TXS).
- Generierung von bis zu 4 frei einstellbaren oder per Skript-Steuerung automatisch steuerbaren analogen Signalen als Eingangssignale in das Modul 1 des AnTeS (TXS).
- Direkter Betrieb von verfahrenstechnischen Simulationen oder alternativ deren Anbindung über das eigene AnTeS-Netzwerk.

- Anbindung beliebiger Signalquellen und -ziele über das eigene AnTeS-Netzwerk. Hierdurch können problemlos auch räumlich getrennt untergebrachte beliebige Systeme ans Modul 1 des AnTeS angebunden werden (vergleiche hierzu auch Abschnitt 2.3.4).

2.3.2 SimGen – Simulation verfahrenstechnischer Systeme

Zur flexiblen Entwicklung verfahrenstechnischer Simulationen für das AnTeS wurde bei der GRS die Software SimGen („Simulation Generator“) in der Programmiersprache Python entwickelt. SimGen erlaubt über eine grafische Oberfläche (vgl. Abb. 2.18) die Erstellung, den Betrieb und die Anbindung verfahrenstechnischer Simulationen, wie z. B. einer BE-Lagerbeckenkühlung oder eines generischen Poolreaktors, an die Leittechniksysteme innerhalb des AnTeS. SimGen ist auf den Mikrorechnern der AnTeS-Interfaces installiert. Darüber hinaus kann SimGen auf jedem Windows-PC installiert werden.

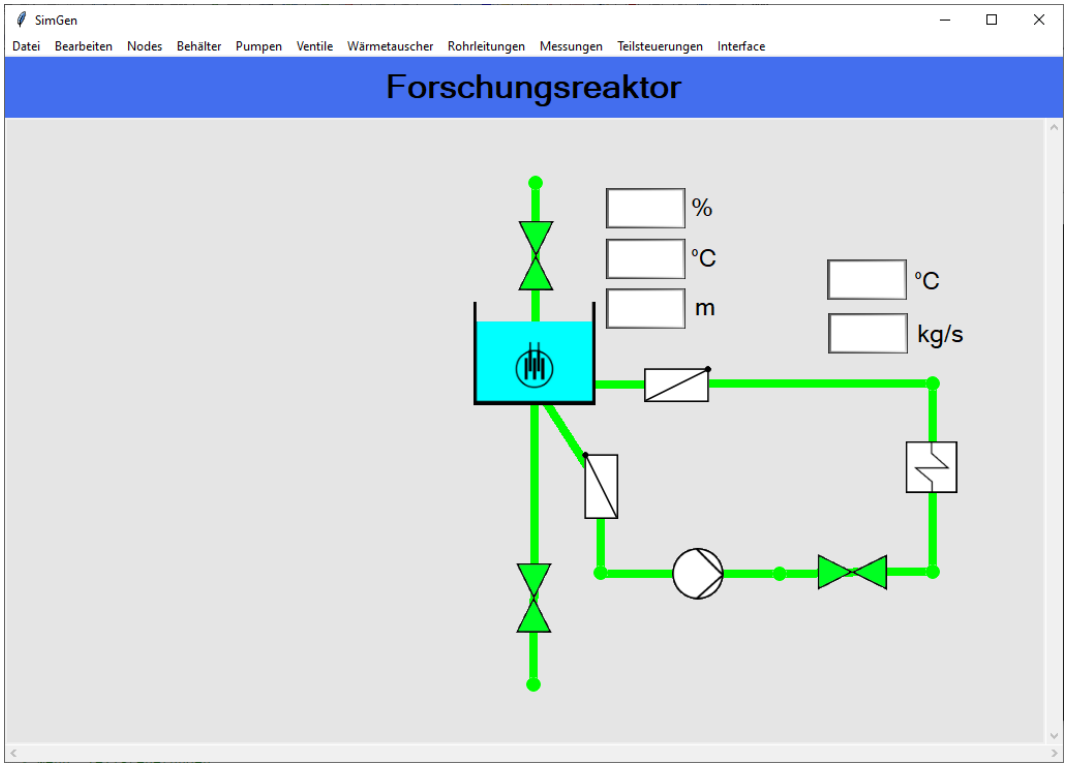


Abb. 2.18 Die Software SimGen der GRS zur Erstellung verfahrenstechnischer Simulationen

Zum aktuellen Zeitpunkt stehen in SimGen die folgenden Elemente für die Erstellung von Simulationen zur Verfügung:

- Behälter
  - Standard-Behälter
  - BE-Becken
  - Pool-Reaktoren
- Pumpen
- Ventile
- Rückschlagklappen
- Wärmetauscher
- Rohrleitungen
- Messungen
  - Druck
  - Füllstand
  - Durchfluss
  - Temperatur
  - Leistung
- Teilsteuerungen

Eine detailliertere Beschreibung der Arbeitsweise und Handhabung von SimGen befindet sich im Anhang A.4. Die Anwendung von SimGen im Rahmen von Tests wird im Abschnitt 4.3.1 beschrieben.

2.3.3 KranSim – Simulation von Maschinenhauskränen

Digitale Leittechnik wird in kerntechnischen Anlagen nicht nur für die Steuerung und Überwachung verfahrenstechnischer Systeme eingesetzt, sondern auch für Hebezeuge. Dort werden viele Abläufe (betriebliche, aber auch sicherheitstechnische Funktionen) automatisiert und auf der Basis digitaler Leittechnik realisiert.

Neben Simulationen wasserführender Systeme (siehe im vorherigen Abschnitt) steht für das AnTeS daher auch eine Simulation eines Maschinenhauskrans zur Verfügung.

Hierfür wurde die Software KranSim („Kransimulation“) in der Programmiersprache Python erstellt (vgl. Abb. 2.19). Die für KranSim benötigten 3D-Modelle des Maschinenhauses und des Krans selbst wurden mit der frei verfügbaren Software Blender /BLE 21/ erstellt. Die Visualisierung des dynamischen Modells erfolgt in KranSim mit Hilfe der 3D-Engine Panda3D /PAN 21 Die Simulation basiert auf bei der GRS vorhandenen Informationen zu entsprechenden realen Kränen in deutschen Kernkraftwerken.

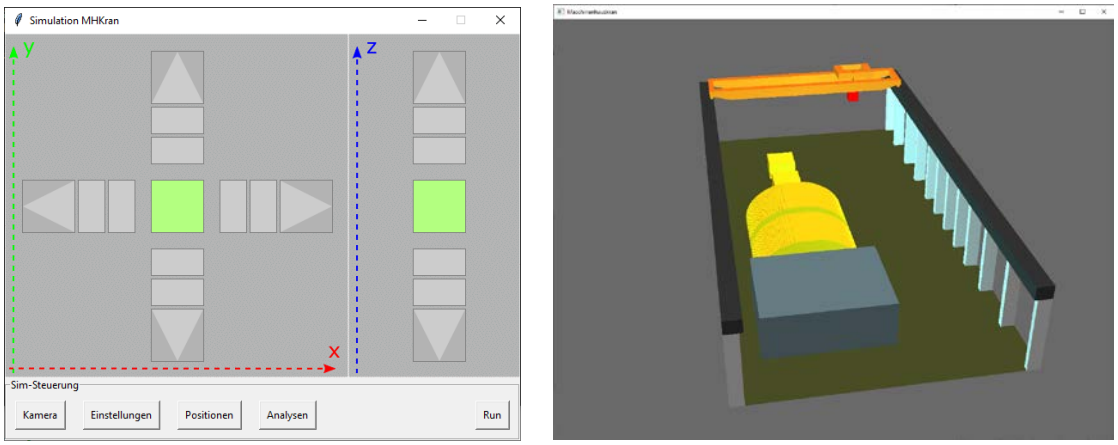


Abb. 2.19 Zwei Detailbilder von der Software KranSim, die von der GRS zur Simulation eines Maschinenhauskrans erstellt wurde.

Eine ausführliche Beschreibung der Arbeitsweise und Handhabung von KranSim befindet sich im Anhang A.5. Die Analysemöglichkeiten in KranSim umfassen allgemeine und Lastabsturzanalysen innerhalb des gesamten Bewegungsraums sowie Analysen entlang eines frei definierbaren Bewegungspfads des simulierten Krans. Die bei automatischen Analysen durchgeführten Kranbewegungen erfolgen wesentlich schneller als in Echtzeit. Eine genauere Beschreibung dieser Analysemöglichkeiten erfolgt im Abschnitt 4.3.2 im Rahmen der Beschreibung der mit dem AnTeS durchgeführten Tests.

2.3.4 TeSys – Testsystem für reale verfahrenstechnische Systeme

Kernkraftwerke bestehen aus sehr vielen Systemen und Komponenten. Über die Leittechnik werden die Prozesse im Kernkraftwerk gesteuert und in der Leitwarte visualisiert. Leittechnik steuert und überwacht in einem Kernkraftwerk verschiedene verfahrenstechnische Komponenten unter unterschiedlichen Betriebsbedingungen und befindet sich in der Realität in Wechselwirkung mit vielen, teilweise unbekannten Einflussfaktoren (z. B. Zeitverzögerungen bei Rückmeldungen, Ungenauigkeiten der Messsignale, Signallaufzeiten).

In der modellbasierten Analyse werden hierfür üblicherweise Annahmen bzw. Vereinfachungen getroffen. Für die Validierung und Verifizierung dieser Annahmen sind experimentelle Untersuchungen erforderlich. In einem ersten Schritt wurden daher Testfeldversuche mit einem vereinfachen Versuchsaufbau durchgeführt.

Um die Anbindung realer Verfahrenstechnik über die Interfaces an das AnTeS zu erproben, wurde im Vorhaben zusätzlich ein Testsystem (TeSys) aus verfahrenstechnischen Komponenten (u. a. Stellventile, Antriebe, Sensoren) aufgebaut. Das TeSys stellt dabei auch eine Möglichkeit dar, das reale Verhalten von Komponenten (z. B. bei der Ansteuerung von Ventilen) im Zusammenspiel mit den Leittechniksystemen vom AnTeS (Module 1 und 2) genauer zu untersuchen bzw. das simulierte Verhalten entsprechender Komponenten in verfahrenstechnischen Simulationen (vgl. Abschnitt 2.3.2) mit der Realität abzugleichen.

Die für das AnTeS entwickelten Interfaces erfüllen eine Reihe verschiedener Aufgaben (siehe Abschnitt 2.3.1). Zum einen können über eine eigens entwickelte Software für das reale Leittechniksystem geeignete Eingangssignale erzeugt und vom Leittechniksystem ausgegebene Signale aufgezeichnet werden, zum anderen können auch Simulationen direkt auf dem Mikrorechnern der AnTeS-Interfaces gestartet werden, welche dann über die entsprechenden Optokoppler und Relais mit dem Leittechniksystem kommunizieren können (siehe z. B. Abschnitt 2.3.2). Zusätzlich sollen die Interfaces aber auch beliebige Signale, beispielsweise mit realen Komponenten, austauschen können. Nicht zuletzt aus baulichen Gründen befinden sich Systeme, mit denen das Leittechniksystem kommunizieren soll, aber ggfs. räumlich entfernt vom Modul 1 des AnTeS. Anstatt in jedem denkbaren Fall eine Vielzahl von Einzelverbindungen (für jedes Ein- und Ausgangssignal ein eigenes Kabel) herzustellen, können beliebige Signale über ein eigenes, internes AnTeS-Netzwerk (Protokoll: Ethernet) ausgetauscht werden, an welches die AnTeS-Interfaces angebunden sind. Diese Vorgehensweise wurde mit Hilfe des nachfolgend beschriebenen Aufbaus entwickelt und getestet.

Der konkrete Testaufbau bestand aus bei der GRS vorhandenen realen Systemkomponenten, um damit die dynamischen Eigenschaften eines verfahrenstechnischen Systems und dessen Wechselwirkungen mit den Modulen von AnTeS in Echtzeit zu untersuchen. Als charakteristisches Beispiel wurde hierzu die Steuerung eines Stellventils in einem Wasserkreislauf durch Leittechnikfunktionen realisiert, wobei u. a. auch die Rückmeldungen des Stellventils berücksichtigt wurden.

Im Rahmen zukünftiger Arbeiten bei der GRS soll das TeSys kontinuierlich weiterentwickelt werden. Im Rahmen dieses Vorhabens wurde zeitweilig ein einfacher Prototyp dieses Testsystems aufgebaut und erfolgreich betrieben. Den schematischen Aufbau dieses Prototyps zeigt Abb. 2.20.

Der TeSys-Prototyp umfasste dabei die folgenden relevanten Bereiche:

- Verfahrenstechnik (Wasserkreislauf),
- Energieversorgung (Schaltanlage) und separate Antriebssteuerung und
- Instrumentierung und Steuerungsfunktionen (Leittechnik).

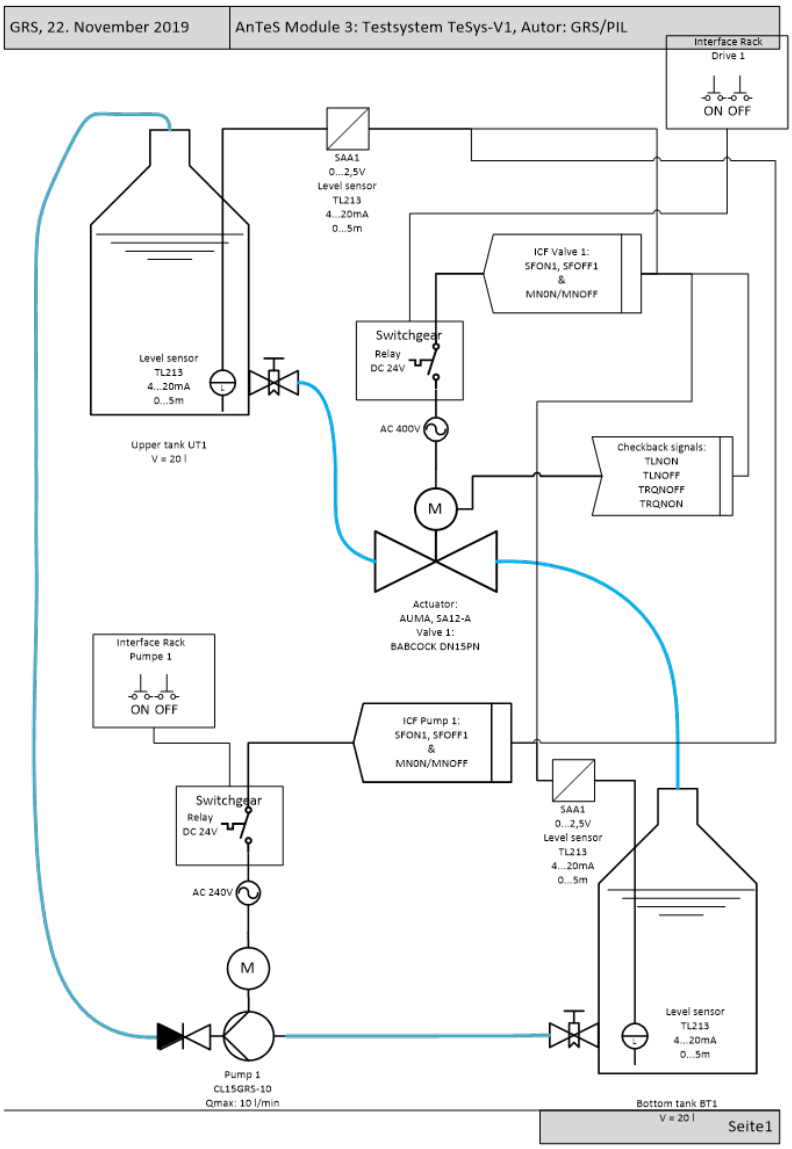


Abb. 2.20 Übersichtsschaltbild des TeSys-Prototyps

2.3.4.1 Verfahrenstechnik

Der Wasserkreislauf des TeSys-Prototyps wurde einsträngig aufgebaut (vgl. Abb. 2.20) und bestand aus den folgenden Komponenten:

- Behälter UT1 („Upper Tank 1“) mit einem Entnahmehahn und Kapazität von ca. 20 l
  - Aufstellhöhe: 209 cm über dem Boden
  - Maximaler Füllstand: 40 cm
- Behälter BT1 („Bottom Tank 1“) mit einem Entnahmehahn und Kapazität von ca. 20 l
  - Aufstellhöhe: 16 cm über dem Boden
  - Maximaler Füllstand: 40 cm
- Förderpumpe Pump 1 mit einem Rückschlagventil auf der Druckseite
- Stellventil Valve 1 motorgesteuert mit einem Handrad (weitere Daten: siehe Doku zu Antrieb AUMA SA12-A und Babcock-Ventil DN15PN/NICO300)
- Verbindungsschläuche: 1/2- und 3/4-Zoll

Elektrische Stellantriebe von Armaturen in Kernkraftwerken regeln den Fluss von Medien durch die Rohrsysteme. Im TeSys-Prototyp wurde ein AUMA-Stellantrieb eingesetzt, dessen Funktionsweise einem typischen motorgesteuerten Ventil in einem generischen Kernkraftwerk entspricht. Dieser wurde durch Leittechnik-Signale gesteuert, wobei die Rückmeldungen des Antriebs (u. a. Wegendschalter TLNON/TLNOFF und DREHMO-Abschaltung TRQNON/TRQNOFF) in den Leittechnik-Funktionen berücksichtigt wurden. Diese leittechnischen Funktionen (siehe Abschnitt 2.3.4.3) hatten die Aufgabe, dass die Füllstände in beiden Tanks jederzeit im erlaubten Bereich bleiben (um z. B. ein vollständiges Entleeren des oberen Tanks und ein damit verbundenes Trockenlaufen der Pumpe zu verhindern).

2.3.4.2 Energieversorgung (Schaltanlage) und separate Antriebssteuerung

Die Stromversorgung des Ventilantriebs erfolgte durch einen Schaltschrank mit einer Steuertafel für vier Motorantriebe (und vier 400-V-Drehstromanschlüssen). Die Ansteuerung des TeSys-Prototyps erfolgte durch eigens installierte Koppelrelais in einem der vier Stromkreise dieses Schaltschranks. Die für den Aufbau des TeSys-Prototyps verwendete Pumpe wird über das 220-V-Netz versorgt, deren Ansteuerung erfolgte ebenfalls über im Schaltschrank installierte Koppelrelais.

Abb. 2.21 Schaltschrank, Antrieb mit angeschlossenem Ventil und Antriebssteuerungsrack vom TeSys

Abb. 2.21 zeigt im linken Bild den für das TeSys verwendeten Schaltschrank. Die Steuertafel enthält die Betriebsartschalter (Hand, Automatik), die Steuerschalter (AUF, ZU), einen Notaus-Schalter sowie Meldeleuchten. In der Betriebsart Automatik kann der Ventilantrieb durch Leittechniksignale aus den Modulen 1 oder 2 gesteuert werden. Im rechten Bild dieser Abbildung ist der verwendete Antrieb (mit unten angeschlossenem Ventil) sowie auf der rechten Seite ein Rack im TeSys-Interface-Schrank mit einer eigens aufgebauten, separaten Antriebssteuerung zu sehen. Diese separate Antriebssteuerung erlaubt eine von der angebundenen Leittechnik unabhängige Steuerung des Antriebs und wurde speziell zu Testzwecken bei der GRS aufgebaut. Damit kann nicht nur der Antrieb gesteuert, sondern auch die Rückmeldungen („Checkbacks“) des Antriebs (z. B. der Wegendschalter und Drehmomentschalter) erfasst und durch Lampen signalisiert werden. Zusätzlich enthält dieses Rack aber auch noch einen (im Einschub der Antriebssteuerung) verbauten Mikrorechner, der einerseits die Rückmeldesignale des Antriebs sowie angeschlossener Messsignalgeber digitalisiert, andererseits aber auch Steuersignale an die Relais im Schaltschrank ausgeben kann. Dieser Mikrorechner kommuniziert über das AnTeS-Netzwerk mit den AnTeS-Interfaces, wodurch beliebige Informationen mit dem Modul 1 von AnTeS ausgetauscht werden können. Durch die Nutzung der Interfaces können beliebige Signale auf einfache Weise mit dem Modul 1 von AnTeS (theoretisch von überall) ausgetauscht werden.

2.3.4.3 Instrumentierung und Steuerungsfunktionen (Leittechnik)

Die Füllstände in den beiden Behältern wurden mittels zweier Füllstandssensoren (Drucksensoren, Messbereich 0 – 500 cm, Ausgangssignal 4…20mA) gemessen und über den im vorhergehenden Abschnitt erwähnten Mikrorechner im Antriebssteuerungseinschub an eines der AnTeS-Interfaces angebunden. Hierdurch standen diese Messwerte im Modul 1 des AnTeS für die Verarbeitung in Funktionsplänen zur Verfügung. Die für die Durchführung von Tests verwendeten leittechnischen Funktionen sind in Abb. 2.22 und Abb. 2.23 dargestellt.

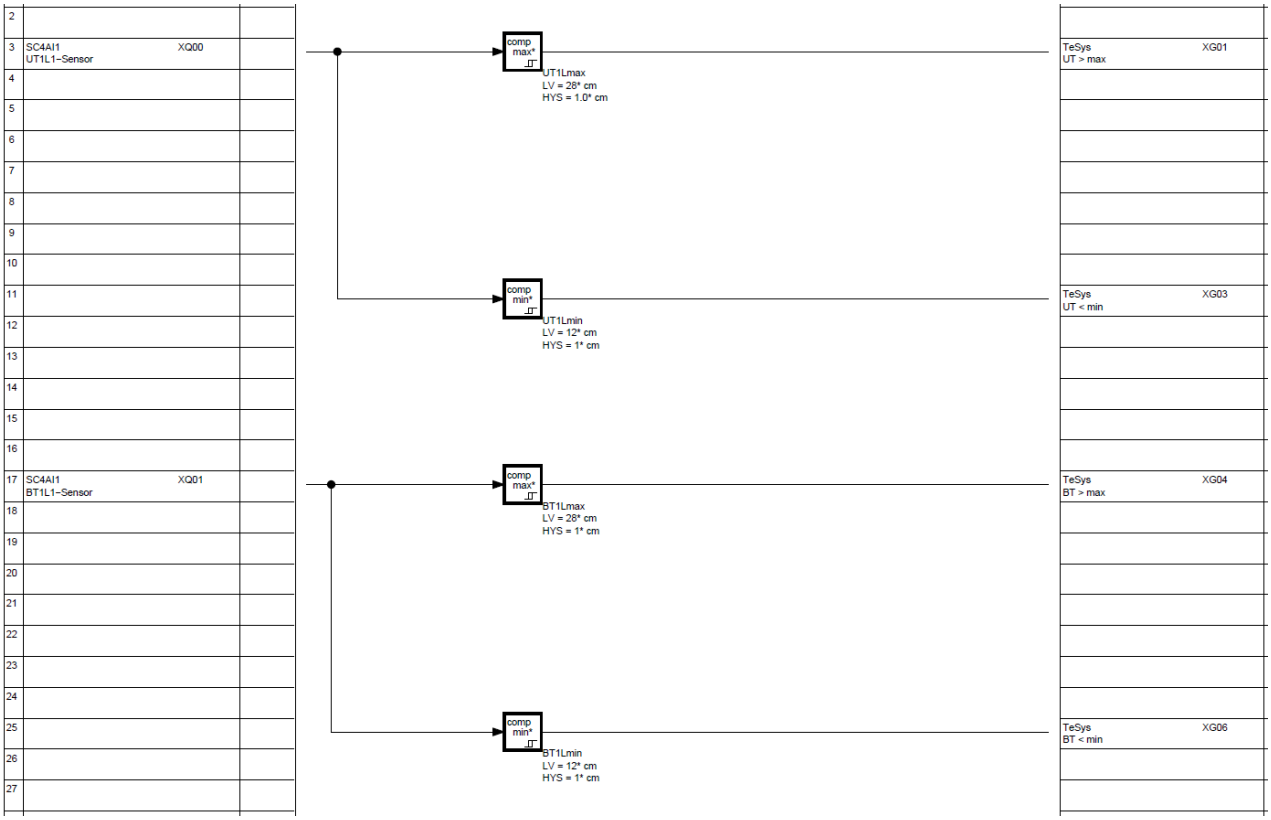


Abb. 2.22 Funktionsplan zur Bildung von Grenzwerten für Tests mit dem TeSys.

Auf dem in Abb. 2.22 gezeigtem Funktionsplan werden die beiden Messsignale der Füllstandsmessungen mit jeweils zwei Grenzwerten („max“, „min“) verglichen und daraus die folgenden vier (binären) Signale gebildet:

- „UT > max“ (UT – „Upper Tank“, oberer Behälter)
  - Spricht an, wenn der Füllstand im oberen Behälter die obere Grenze erreicht.
  - Grenzwert: 28 cm
- „UT < min“ (UT – „Upper Tank“, oberer Behälter)
  - Spricht an, wenn der Füllstand im oberen Behälter die untere Grenze erreicht.
  - Grenzwert: 12 cm
- „BT > max“ (BT – „Bottom Tank“, unterer Behälter)
  - Spricht an, wenn der Füllstand im unteren Behälter die obere Grenze erreicht.
  - Grenzwert: 28 cm
- „BT < min“ (BT – „Bottom Tank“, unterer Behälter)
  - Spricht an, wenn der Füllstand im unteren Behälter die untere Grenze erreicht.
  - Grenzwert: 12 cm

Abb. 2.23 zeigt, wie aus diesen Grenzwertsignalen für Testzwecke Auslösesignale gebildet wurden. In der oberen Hälfte des Funktionsplans wird beim gleichzeitigen Anstehen (über mindestens 0,5 s) von „BT > max“ und „UT < min“ die Pumpe gestartet und das Ventil vom TeSys geschlossen. Hierdurch wird das Wasser im System aus dem offensichtlich gefüllten unteren Behälter in den oberen Behälter gepumpt. Umgekehrt wird die Pumpe gestoppt und das Ventil geöffnet, falls die Grenzwerte „BT < min“ und „UT > max“ gleichzeitig für mehr als 0,5 s anstehen (untere Hälfte des Funktionsplans). In diesem Fall entleert sich der obere Behälter in den unteren Behälter über das geöffnete Ventil.

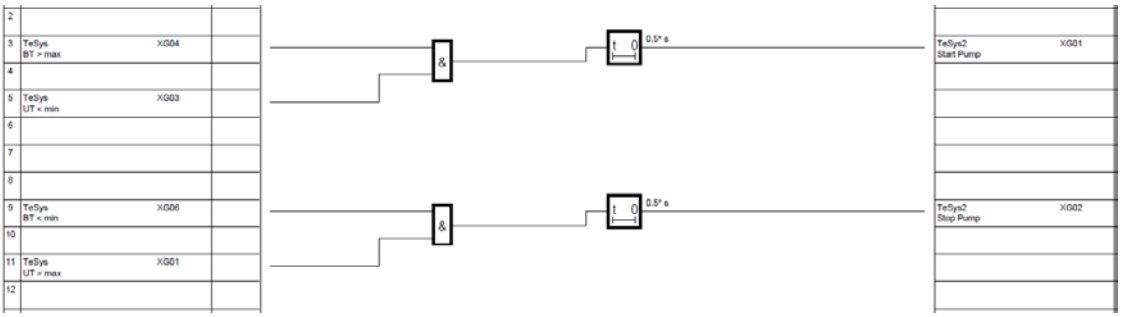


Abb. 2.23 Funktionsplan für die Bildung von Auslösesignalen bei Tests mit TeSys

2.3.4.4 Inbetriebnahme und erste Tests

Mit dem in den vorangegangenen Abschnitten beschriebenen Aufbau wurden unterschiedliche Tests durchgeführt. Hierbei wurden insbesondere die Grenzwerte bewusst angefahren und das zeitliche Verhalten des Gesamtsystems aufgezeichnet.

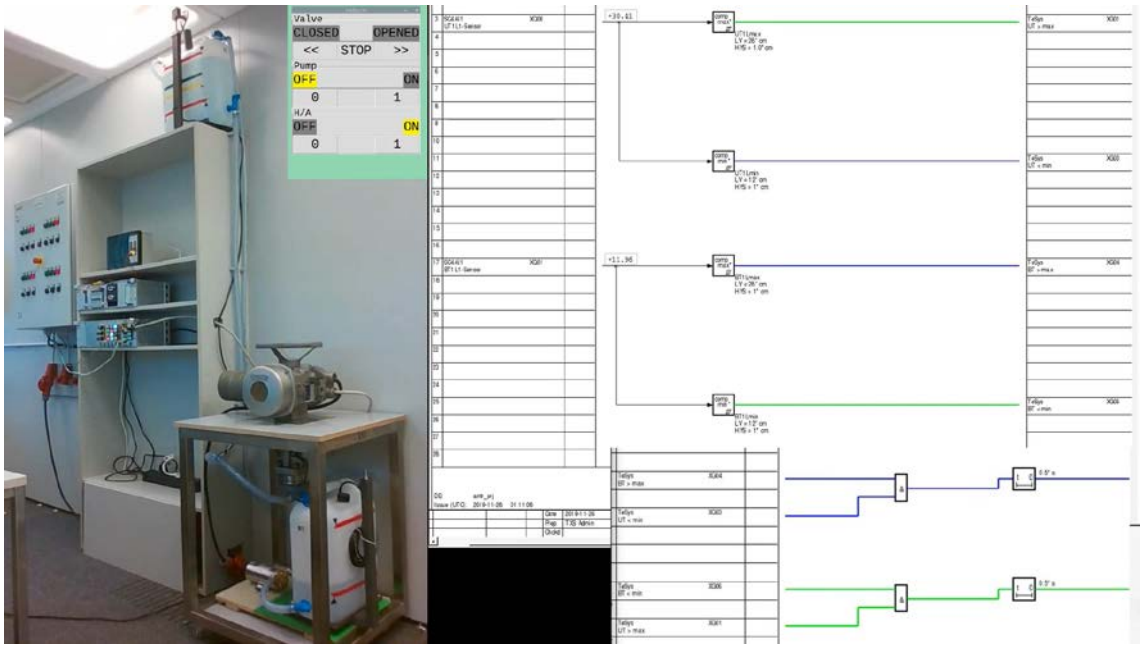


Abb. 2.24 TeSys in Betrieb

Abb. 2.24 zeigt exemplarisch eine typische Situation beim Ansprechen von Grenzwerten im TeSys-Prototyp. Der obere Behälter im linken Teil des Bildes ist bis zur oberen Grenze (durch oberen roten Strich gekennzeichnet) gefüllt, der untere Behälter ist in diesem Bild bis zur unteren Grenze geleert (unterer roter Strich). Dementsprechend stehen rechts im Bild im oberen (dynamischen) Funktionsplan die beiden Grenzwerte „UT > max“ und „BT < min“ an (grün gekennzeichnet), die beiden weiteren Grenzwerte („UT < min“ und „BT > max“) nicht (blau gekennzeichnet). Aus den beiden anstehenden Grenzwerten wird im unteren Funktionsplan im Bild das Ansteuerungssignal zum Stoppen der Pumpe und dem Öffnen des Ventils gebildet (Anmerkung: in der gezeigten Momentaufnahme öffnet sich das Ventil gerade, dementsprechend ist das Ventil gerade weder ganz offen noch geschlossen; außerdem ist schon ein kleiner Teil des Wassers wieder in den unteren Behälter zurückgeflossen).

Die mit dem TeSys durchgeführten Tests zeigten durchgängig das erwartete Verhalten des Gesamtsystems. Insbesondere ergaben sich durch die Anbindung des Systems an das Modul 1 des AnTeS über den Umweg über das AnTeS-Netzwerk (Ethernet) keine signifikanten Verzögerungen. Mit Hilfe des TeSys-Prototyps konnte also gezeigt werden, dass prinzipiell die Anbindung beliebiger Signalquellen und -ziele an das AnTeS über die Interfaces problemlos umsetzbar ist und insbesondere auch entfernte Systeme angebunden werden können.

3 Werkzeug- und Methodenentwicklungen

Ein wesentliches Ziel des Vorhabens war die Weiterentwicklung und der Test von Methoden zur Analyse der Zuverlässigkeit und Sicherheit digitaler Leittechniksysteme, u. a. durch Auswirkungsanalysen von Fehlerkombinationen und Sensitivitätsanalysen. In diesem Kapitel werden daher die im Rahmen des aktuellen Vorhabens neu entwickelten Werkzeuge und Methoden vorgestellt, welche für Analysen und Tests eingesetzt wurden.

3.1 Fehlerinjektion im AnTeS

Bei der Verwendung des AnTeS ist ein wesentlicher Aspekt, dass Fehler/Ausfälle in den Leittechniksystemen (Module 1 und 2) gezielt durch eine sogenannte Fehlerinjektion aktiviert oder deaktiviert werden können. Bevor nachfolgend die für das AnTeS entwickelte Methode zur Fehlerinjektion vorgestellt wird, werden für eine eindeutige (und somit auch besser verständliche) Darstellung, die in diesem Abschnitt gültigen Begriffskonventionen dargelegt:

„Fehler“:
Fehler (bzw. Fault) beschreibt einen einzelnen (postulierten) Ausfall eines Signals bzw. eines Funktionsblocks oder einer Baugruppe. Diese Ausfälle können sich in ihrer Fehlerausfallart (und damit in ihrer Auswirkung auf Leittechnikfunktionen) unterscheiden.

„Fehlersignal“:
Fehlersignale (Fault-Signals) dienen der (An-)Steuerung von Fehlern bei Anwendung der Fehlerinjektion. Es kann sich um den Ausgangswert eines Simulink-Speicherbausteins (Funktionsblock „Data Store“ im Modul 2 des AnTeS), um ein einzelnes in das Modul 1 eingespeistes Signal oder um das Ausgangssignal eines Speicherbaustein-Funktionsblocks (RS-Flipflop-Funktionsblock) des TXS handeln. Im letzten Fall werden diese Speicherbausteine über eine Kombination von Eingangssignalen (Fehleradresse) über das Interface des AnTeS gezielt gesetzt oder gelöscht.

„Fehleradresse“:

Eine Fehleradresse ist eine einzustellende Kombination von Eingangssignalen ins TXS (über das Interface des AnTeS), um einen bestimmten Fehler innerhalb des Moduls 1 zu steuern. Ein zusätzliches Eingangssignal in das TXS-System bestimmt hierbei, ob der adressierte Fehler gesetzt oder gelöscht wird.

„Funktionsblock“:

Funktionsblöcke dienen in den Modulen 1 und 2 des AnTeS der grafischen Umsetzung von Leittechnikfunktionen. Beispiele für Funktionsblöcke sind UND-Bausteine oder 2.MAX-Bausteine (siehe hierzu auch /TXS 12/).

„Baugruppe“:

Reale (Modul 1 des AnTeS) oder simulierte (Modul 2 des AnTeS) Hardware, typischerweise in Form von Karten, die in Racks des Leittechniksystem geschoben werden können (z. B. SVE2 (Prozessorkarte) oder SDO1 (Digitale Ausgangskarte) im TXS). Es kann sich aber auch um andere Hardware (z. B. das Rack selbst) handeln.

„Fehlerinjektion“:

Fehlerinjektion (bzw. Fault-Injection) ist die nachfolgend beschriebene Methode zur gezielten Ansteuerung von einzelnen Fehlern oder Fehlerkombinationen innerhalb des AnTeS. Die im nachfolgenden Abschnitt beschriebene Methode für automatisierte Fehlerauswirkungsanalysen ermöglicht dabei auch eine automatisierte Ansteuerung von Fehlerkombinationen.

„Fehlerkombination“:

Eine Kombination aus aktiven und inaktiven Fehlern innerhalb eines Testfalls (Test-Case).

„Fehlerausfallarten“:

Die Fehlerausfallart (bzw. Failure-Mode) in einem sicherheitsrelevanten digitalen Leittechniksystem unterscheidet zwischen nicht-selbstmeldenden (NSF) und selbstmeldenden (SF) Fehlern. Im ersten Fall (NSF) hat ein Signal (z. B. das Ausgangssignal aus einem Funktionsblock) unentdeckt einen falschen Wert. Im zweiten Fall liegt ebenfalls ein fehlerhafter Wert vor, dieser ist aber (innerhalb des Leittechniksystems) als fehlerhaft markiert (d. h., dass das Fehlerattribut dieses Signals gesetzt ist).

„Fehlerattribut“:

Das Fehlerattribut (Error-Attribut) ist eine jedem Signal innerhalb des digitalen Leittechniksystems angehängte Information („Flag“), die dieses Signal als valide oder nicht-valide kennzeichnet. Erkannte Fehler (SF) können bei der Verarbeitung von Signalen innerhalb von Funktionsblöcken durch Auswertung des Fehlerattributs berücksichtigt werden. Valide[2] Signale innerhalb eines digitalen Leittechniksystems können also entweder korrekt oder nicht-selbstmeldend (NSF) ausgefallen sein.

Für verschiedene Zielstellungen in unterschiedlichen Analysen mit dem AnTeS müssen Fehler unterschiedlicher Fehlerausfallarten (Failure-Modes) im jeweils betrachteten Leittechniksystem ein- oder abgeschaltet werden können. Diese Fehler werden über eigens dafür vorgesehene externe Signale gesteuert (Fehlersignale bzw. Fault-Signals). Grundsätzlich lassen sich Fehlerausfallarten in einem sicherheitsrelevanten digitalen Leittechniksystem in nichtselbstmeldende (NSF) und selbstmeldende Fehler (SF) aufteilen. Im ersten Fall hat ein Signal (z. B. das Ausgangssignal aus einem Funktionsblock) unentdeckt einen falschen Wert. Im zweiten Fall liegt ebenfalls ein fehlerhafter Wert vor, dieser ist aber (innerhalb des Leittechniksystems) als fehlerhaft markiert (d. h., dass das Fehlerattribut dieses Signals gesetzt ist).

Abb. 3.1 zeigt, wie ein Signal („Digitaler Eingang 1“) in einem Funktionsplan gezielt im Sinne einer Fehlerinjektion beeinflusst werden kann:

- Sowohl das Signal „Fehler 1“ als auch das Signal „Fehler 3“ schalten über das ODER (d) den Binärschalter (a) auf den festen („falschen“) Wert „0“ (b).
- Das Signal „Fehler 3“ bewirkt zusätzlich, dass das Signal über den Fehlersetzer (TXS-Funktionsblock FSFB, c) ein Fehlerattribut erhält.
- In diesem Sinne bewirkt somit „Fehler 1“ einen nicht-selbstmeldenden Ausfall (NSF) des Signals „Digitaler Eingang 1“, „Fehler 3“ löst einen selbstmeldenden Ausfall (SF) des Signals „Digitaler Eingang 1“ aus.

[2] „Valide“ meint in diesem Zusammenhang also nicht, dass ein Signal fehlerfrei ist. Stattdessen wird das Signal vom Leittechniksystem für korrekt gehalten, da das Fehlerattribut nicht gesetzt ist. Ein solches Signal kann entweder tatsächlich fehlerfrei oder unerkannt falsch sein (nicht-selbstmeldender Fehler).

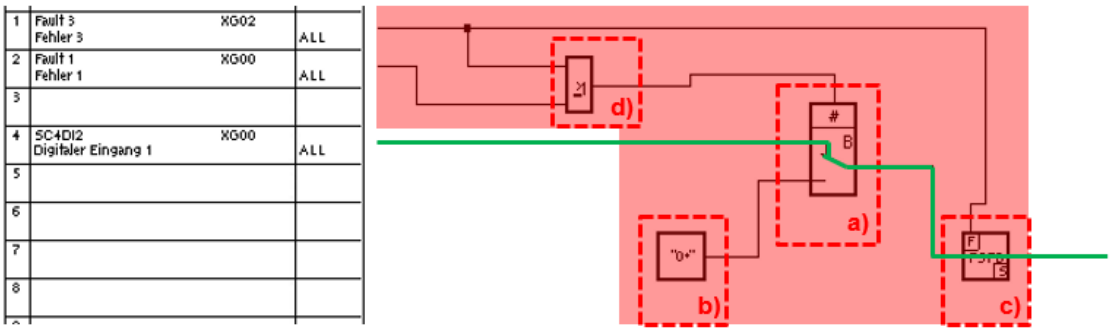


Abb. 3.1 Gezielte Fehlersteuerung eines Signals in einem Funktionsplan. Der ungestörte Signalverlauf ist grün gekennzeichnet.

Die Anwendung dieser Art der Fehlerinjektion wird im Folgenden an einem einfachen Beispiel weiter ausgeführt. Betrachtet wird ein sehr einfacher Funktionsplan, der nur einen einzigen Funktionsblock enthält, in diesem Fall ein logisches UND mit zwei Eingängen. Dementsprechend verfügt dieser Funktionsplan also über zwei Eingänge, einen Ausgang und einen UND-Funktionsblock. Hinzu kommen weitere Funktionsblöcke, die ausschließlich für die Fehlerinjektion sowie das Auslesen des Fehlerattributs des Ausgangs des UND-Funktionsblocks benutzt werden. Inklusive Logik für die Fehlerinjektion sieht der entsprechende Funktionsplan im TXS-System dann wie in Abb. 3.2 dargestellt aus.

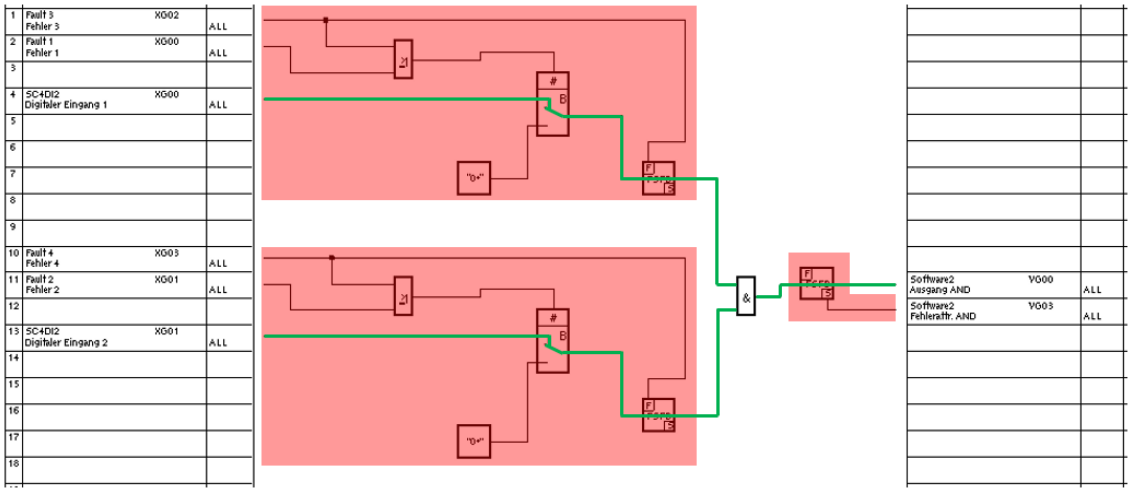


Abb. 3.2 Funktionsplan zur Untersuchung des Verhaltens eines UND-Funktionsblocks.

Die rot gekennzeichneten Bereiche dienen ausschließlich der Fehlerinjektion und dem Auslesen der Fehlerauswirkungen. Der „normale“ Signalverlauf ist grün gekennzeichnet.

Bereits für die Analyse eines verhältnismäßig einfachen Funktionsplans (mit nur einem UND-Funktionsblock) werden also bereits vier Fehlersignale benötigt. Unter Umständen werden also mehr Fehlersignale benötigt als dafür digitale Eingangskanäle in das reale TXS-System verfügbar sind oder benutzt werden können. Daher wird weiter unten eine Methode dargelegt, wie deutlich mehr Fehlersignale gesteuert werden können als Eingangssignale zur Verfügung stehen.

Im Modul 2 des AnTeS besteht die Einschränkung nicht, dass evtl. nicht genügend Eingangskanäle zur Verfügung stehen. Hier empfiehlt es sich vielmehr, dass die Fehlersignale über Speicherbausteine („Data Store“) verwaltet werden (vgl. Abb. 3.5). Diese können dann an den entsprechenden Stellen über „Data Store Read“-Bausteine zum Aktivieren oder Deaktivieren von Fehlern im System verwendet werden.

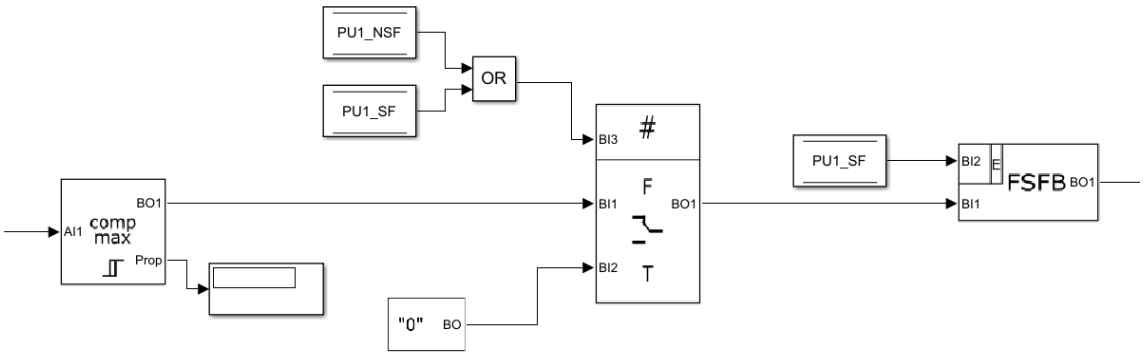


Abb. 3.3 Beispiel für eine Fehlerinjektion in das Ausgangssignal eines Grenzwertbausteins („comp“) im Modul 2 (Matlab/Simulink) des AnTeS.

PU1_SF und PU1_NSF lesen den aktuellen Wert der Fehlersignale über sogenannten „Data Store Read“-Funktionsblöcke aus den entsprechenden Speichern aus (PU1_NSF, PU1_SF).

Anmerkung:
In Abb. 3.5 sehen die verwendeten Funktionsblöcke beinahe genauso aus, wie sie auch im TXS zur Verfügung stehen. Dies liegt daran, dass innerhalb von Modul 2 des AnTeS TXS-Funktionsblöcke nachgebildet wurden. Hinter den scheinbaren TXS-Funktionsblöcken verbergen sich sogenannte Subsysteme von Simulink, die das Verhalten von TXS-Funktionsblöcken 1:1 nachbilden (siehe Abschnitt 2.2).

Fehlerinjektion speziell im Modul 1 des AnTeS

Nachfolgend wird erläutert, wie Fehlersignale bei Fehlerinjektionen speziell innerhalb des Moduls 1 generiert und gesteuert werden können. Diese Beschreibung beschränkt sich dabei auf die im aktuellen Vorhaben verwendete Vorgehensweise (bei den im Kapitel 4 dargestellten Analysen). Im Rahmen des aktuellen Vorhabens wurden darüber hinaus erste Versuche durchgeführt, auf veränderliche Parameter innerhalb des TXS über den Servicerechner direkt zuzugreifen (anstatt, wie hier beschrieben, die AnTeS-Interfaces zu verwenden). Auf diese Weise kann die Fehlerinjektion ins Modul 1 von AnTeS künftig vereinfacht und beschleunigt werden (siehe hierzu Abschnitt A.1 im Anhang).

Für die nachfolgenden Betrachtungen wird davon ausgegangen, dass für die Ansteuerung von Fehlern 8 binäre Eingänge des TXS (über das Interface des AnTeS zu den entsprechenden SDI1-Baugruppen (TX**S** **D**igital **I**nput Module)) zur Verfügung stehen. In diesem Fall können bis zu 128 Fehlersignale gesteuert werden. Prinzipiell können bei N verfügbaren digitalen Eingangskanälen auf die in diesem Kapitel vorgestellte Weise $2^{(N-1)}$ Fehlersignale gesteuert werden (also beispielsweise 32.768 Fehlersignale bei 16 digitalen Eingängen).

Bei der hier vorgestellten Steuerung (Adressierung) von Fehlersignalen werden die digitalen Eingänge nicht 1:1 mit einem bestimmten Fehler für die Fehlerinjektion verknüpft. Stattdessen werden 7 der 8 Kanäle für eine Adressierung und der 8. Kanal für das Aktivieren oder Deaktivieren („0“ oder „1“) des jeweils angesteuerten Fehlers verwendet (die eigentlichen Fehlersignale sind die Ausgänge von Speicherbausteinen). Diese grundsätzliche Idee wird durch Abb. 3.6 verdeutlicht.

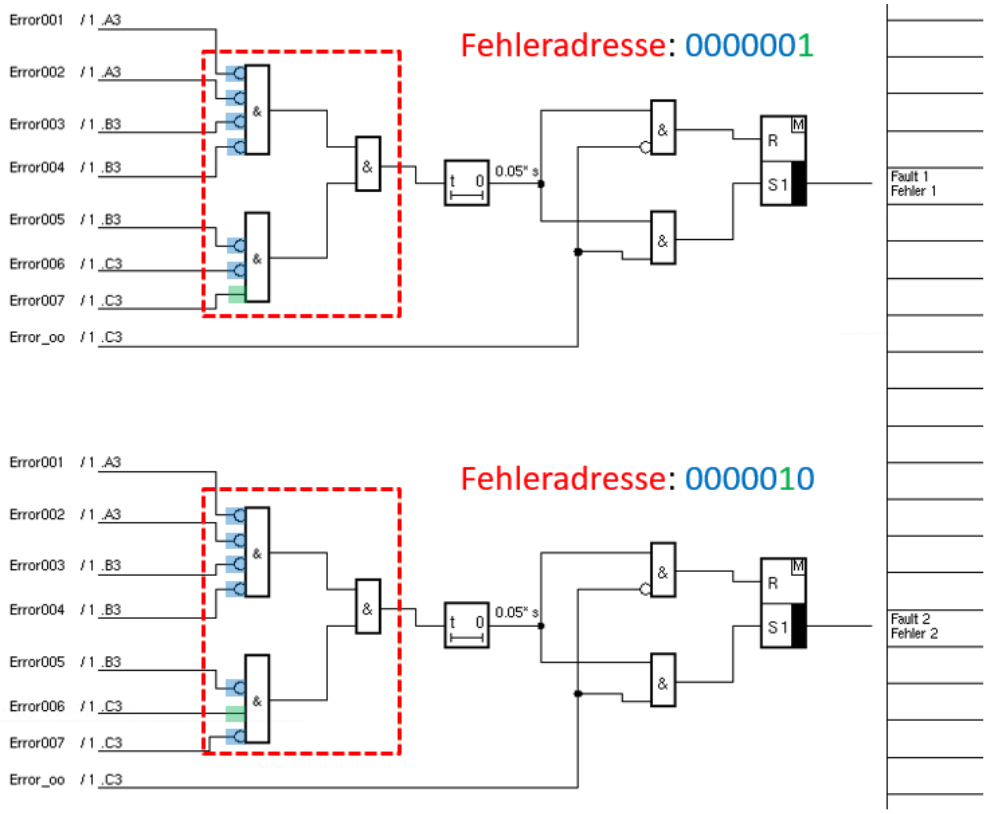


Abb. 3.4 Die Steuerung der ersten beiden von 128 möglichen Fehlersignalen

Auf der linken Seite in dieser Abbildung gehen die ersten 7 (Error001 bis Error007) der insgesamt 8 digitalen Eingänge auf jeweils zwei UND-Funktionsblöcke, die ihrerseits wieder über einen dritten UND-Funktionsblock verknüpft sind. Die drei UND-Funktionsblöcke stellen somit gemeinsam eine UND-Verknüpfung mit insgesamt 7 Eingängen dar (die Verwendung von drei Funktionsblöcken ist im TXS-System notwendig, da UND-Funktionsblöcke hier maximal 4 Eingänge haben können). Die Adressierung eines Fehlersignals findet jetzt über negierte und nicht negierte Eingänge in dieser Logik statt (negierte Eingänge sind durch Kreise am jeweiligen Funktionsbaustein-Eingang erkennbar).

So kann beispielsweise das Fehlersignal „Fehler 1“ („Fault 1“) über die folgende Kombination von Eingangssignalen angesprochen werden:

Digitaler Eingang 1 == Error001 = 0
Digitaler Eingang 2 == Error002 = 0
Digitaler Eingang 3 == Error003 = 0
Digitaler Eingang 4 == Error004 = 0
Digitaler Eingang 5 == Error005 = 0
Digitaler Eingang 6 == Error006 = 0
Digitaler Eingang 7 == Error007 = 1

Um beispielsweise das Fehlersignal „Fehler 2“ („Fault 2“) anzusprechen, ist die folgende Kombination von Eingangssignalen notwendig:

Digitaler Eingang 1 == Error001 = 0
Digitaler Eingang 2 == Error002 = 0
Digitaler Eingang 3 == Error003 = 0
Digitaler Eingang 4 == Error004 = 0
Digitaler Eingang 5 == Error005 = 0
Digitaler Eingang 6 == Error006 = 1
Digitaler Eingang 7 == Error007 = 0

Interpretiert man die jeweils 7 Eingangskanäle gemeinsam als Binärzahl, so wird also „Fehler 1“ über die Binärzahl 0000001 und „Fehler 2“ über die Binärzahl 0000010 angesprochen. Übersetzt in Dezimaldarstellung sind das gerade die beiden Zahlen 1 und 2. Bei entsprechender Verschaltung für jedes einzeln zu steuernde Fehlersignal stehen also Kombinationen zwischen 0000001 (dezimal 1) und 1111111 (dezimal 128) zur Verfügung.

Der achte Eingangskanal („Error_oo“, „oo“ steht hierbei für „on-off“) wird dafür verwendet, den entsprechenden Speicherbaustein (RS-Flipflop-Funktionsblock) des ausgewählten Fehlersignals zu setzen oder zu löschen (über zwei weitere UND-Funktionsblöcke). Dieser Zusammenhang wird durch Abb. 3.5 verdeutlicht.

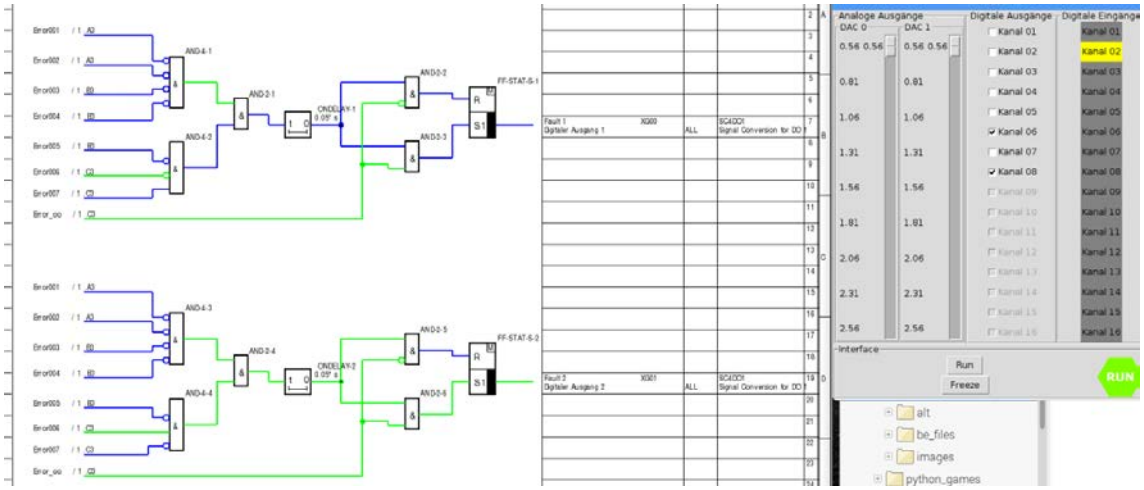


Abb. 3.5 Setzen eines Speicherbaustein (RS-Flipflop-Funktionsblock) des ausgewählten Fehlersignals

Im dargestellten Fall sind die Eingangskanäle 6 und 8 auf „1“ (grün), alle anderen Eingangskanäle auf „0“ (blau) gesetzt. Hierdurch wird das Fehlersignal „Fehler 2“ (unten) auf „1“ (aktiv) gesetzt.

Für diese Abbildung wurden – sichtbar auf der rechten Seite der Abbildung – in der Interface-Steuerung die Kanäle 6 und 8 auf „1“ (gesetzte Häkchen), die anderen Kanäle auf „0“ gesetzt. Auf der linken Seite ist die Fehleradressierung der Fehlersignale „Fehler 1“ und „Fehler 2“ („Fault 1“ und „Fault 2“) als dynamische Funktionspläne (im TXS) dargestellt. Man erkennt, dass für die Eingangskombination 0000010 (Eingangskanäle 1 – 7) das zusammengesetzte UND nur für „Fehler 2“ erfüllt ist. Das 8. Signal sorgt dafür, dass der Speicherbaustein für „Fehler 2“ („Fault 2“) gesetzt wird. Setzt man hingegen das 8. Signal auf „0“, so wird der Speicherbaustein für „Fehler 2“ („Fault 2“) gelöscht (vgl. Abb. 3.8).

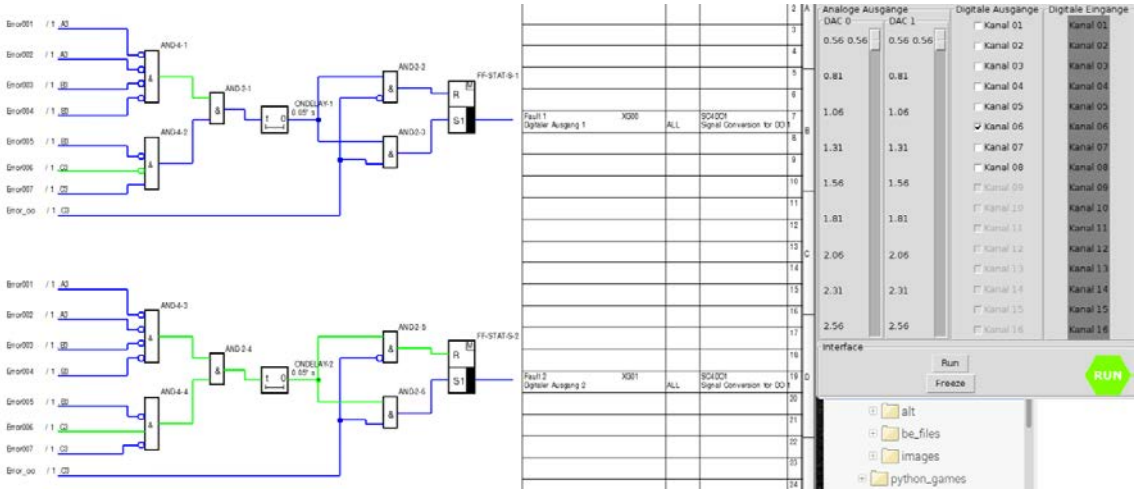


Abb. 3.6 Setzen eines Speicherbaustein (RS-Flipflop-Funktionsblock) des ausgewählten Fehlersignals (inaktiv)

Im Gegensatz zur Situation in Abb. 3.5 ist hier nur der Eingangskanal 8 auf „0“ gesetzt, hierdurch wird „Fehler 2“ gelöscht.

Wie bereits erwähnt, stehen auf diese Weise 128 Fehleradressen zur Verfügung (bei Verwendung von 7 Eingangskanälen und einem Eingang zum Setzen oder Löschen). Jeder zusätzliche Eingang verdoppelt jeweils die Anzahl der möglichen Fehleradressen. Dies stellt einen erheblichen Vorteil dar gegenüber der direkten Verwendung von digitalen Eingangskanälen als Fehlersignale. Demgegenüber steht der Nachteil, dass auf diese Weise immer nur ein Fehler nach dem anderen gesetzt oder gelöscht werden kann. Da diese allerdings über Speicherbausteine dauerhaft gesetzt oder gelöscht werden können, sind dennoch alle Fehlerkombinationen einstellbar. Nur müssen hierzu für jede Fehlerkombination alle Fehlersignale nacheinander gesetzt werden, wozu (auch bei einer automatisierten Ansteuerung) deutlich mehr Zeit benötigt wird.

Ein Funktionsblock in Abb. 3.5 und Abb. 3.6 wurde bisher noch nicht erläutert. Hinter den UND-Funktionsblöcken, die der Adressierung des jeweiligen Fehlers dienen, befindet sich jeweils eine Anzugsverzögerung („ONDELAY“) mit einer eingestellten Verzögerungszeit von 0,05 Sekunden. Diese 0,05 Sekunden entsprechen gerade der eingestellten TXS-Zykluszeit von 50 ms. Hierdurch wird sichergestellt, dass beim Umschalten von der Adresse eines Fehlers auf die Adresse eines anderen Fehlers nicht versehentlich weitere Adressen anderer Fehler angesprochen werden (weil z. B. genau beim Zykluswechsel im TXS-System noch nicht alle Eingänge umgestellt wurden).

3.2 Automatisierte Fehlerauswirkungsanalysen im AnTeS

Die im Abschnitt 3.1 erläuterte Möglichkeit zur Fehlerinjektion erlaubt eine automatisierte Ermittlung der Auswirkungen von postulierten Fehlern (und damit z. B. auch die Validierung von FMEA-Ergebnissen). Die grundsätzliche Idee dahinter ist es, automatisch alle Kombinationen von Ausfällen und Fehlern in die Module 1 oder 2 des AnTeS einzuspeisen und das dabei zu beobachtende Gesamtverhalten des Systems aufzuzeichnen (Auslösung ja/nein?).

Die im vorangegangenen Abschnitt beschriebene Methode zu Steuerung injizierter Fehler in ein Leittechniksystem erfordert für eine automatisierte Fehlerkombinationsauswirkungs-Analyse die wechselnde Einstellung einer Vielzahl von Signalkombinationen. Aus diesem Grund wurde ein Programm entwickelt, das diese Aufgabe vollautomatisch erledigt und gleichzeitig auch die Ergebnisse der jeweiligen Analyse aufzeichnet. Nach einer kurzen Erläuterung der Funktionsweise dieser Software wird anschließend die Einsetzbarkeit dieser Software an einem einfachen Beispiel demonstriert.

Nach dem Start des Programms auf dem Mikrorechner (Raspberry Pi 3 B+) des Interfaces des AnTeS öffnet sich das in Abb. 3.9 dargestellte Fenster. Hier können in drei unterschiedlichen Bereichen Einstellungen vorgenommen werden: Auf der linken Seite („Eingänge (SDI)“) sind Eingangssignale (über SDI-Karten) in das TXS-System dargestellt. Die ersten 8 Kanäle dienen (wie im vorigen Abschnitt erläutert) der Steuerung der Fehlersignale und sind dementsprechend nicht für den Anwender verfügbar. Darunter können aber für die übrigen Eingangssignale feste Werte während eines Analysedurchlaufs ausgewählt werden. Kanäle mit einem Haken werden während der Analyse konstant mit einer logischen „1“ beschaltet, an allen anderen Kanälen liegt eine logische „0“. Im mittleren Bereich („Einstellungen“) wird die Anzahl der Fehlersignale und eine Wartezeit in Millisekunden eingestellt. Für die Anzahl der zu kombinierenden Fehler sind derzeit Werte zwischen 1 und 128 erlaubt. Die Bedeutung der Wartezeit wird weiter unten beschrieben, hier kann aber standardmäßig der voreingestellte Wert von 150 ms beibehalten werden. Auf der rechten Seite („Ausgänge (SDO)“) werden ebenfalls über Haken diejenigen Kanäle ausgewählt, die während der Analyse aufgezeichnet werden sollen.

Die eigentliche Analyse wird durch Betätigen des Schalters „Start“ in Gang gesetzt.

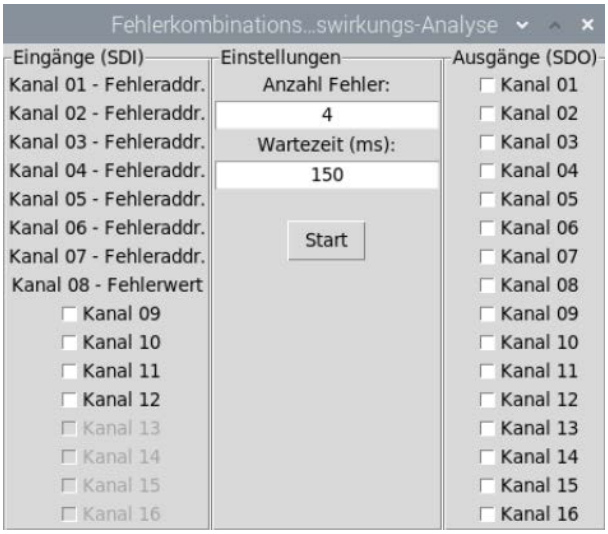


Abb. 3.7 Eingabefenster des Programms „Fehlerkombinationsauswirkungs-Analyse“

Dargestellt auf dem Raspberry Pi 3 B+ des Interfaces des AnTeS.

Nach dem Start der Analyse werden nacheinander alle Fehlerkombinationen, die sich aus der Anzahl der Fehlersignale ergeben, nacheinander in das TXS-System eingespeist. Wählt man beispielsweise für die Anzahl der Fehler 6, so werden nacheinander die in Tab. 3.1 aufgelisteten Fehlerkombinationen eingestellt.

Tab. 3.1 Sämtliche Fehlerkombinationen bei einer Einstellung von 6 Fehlern im Programm „Fehlerkombinationsauswirkungs-Analyse“

#	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6
1	0	0	0	0	0	0
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	0	0	0	1	1
5	0	0	0	1	0	0
6	0	0	0	1	0	1
7	0	0	0	1	1	0
8	0	0	0	1	1	1
9	0	0	1	0	0	0
10	0	0	1	0	0	1
11	0	0	1	0	1	0
12	0	0	1	0	1	1
13	0	0	1	1	0	0
14	0	0	1	1	0	1
15	0	0	1	1	1	0
16	0	0	1	1	1	1
17	0	1	0	0	0	0
18	0	1	0	0	0	1

#	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6
19	0	1	0	0	1	0
20	0	1	0	0	1	1
21	0	1	0	1	0	0
22	0	1	0	1	0	1
23	0	1	0	1	1	0
24	0	1	0	1	1	1
25	0	1	1	0	0	0
26	0	1	1	0	0	1
27	0	1	1	0	1	0
28	0	1	1	0	1	1
29	0	1	1	1	0	0
30	0	1	1	1	0	1
31	0	1	1	1	1	0
32	0	1	1	1	1	1
33	1	0	0	0	0	0
34	1	0	0	0	0	1
35	1	0	0	0	1	0
36	1	0	0	0	1	1
37	1	0	0	1	0	0
38	1	0	0	1	0	1
39	1	0	0	1	1	0
40	1	0	0	1	1	1
41	1	0	1	0	0	0
42	1	0	1	0	0	1
43	1	0	1	0	1	0
44	1	0	1	0	1	1
45	1	0	1	1	0	0
46	1	0	1	1	0	1
47	1	0	1	1	1	0
48	1	0	1	1	1	1
49	1	1	0	0	0	0
50	1	1	0	0	0	1
51	1	1	0	0	1	0
52	1	1	0	0	1	1
53	1	1	0	1	0	0
54	1	1	0	1	0	1
55	1	1	0	1	1	0
56	1	1	0	1	1	1
57	1	1	1	0	0	0
58	1	1	1	0	0	1
59	1	1	1	0	1	0
60	1	1	1	0	1	1
61	1	1	1	1	0	0

#	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6
62	1	1	1	1	0	1
63	1	1	1	1	1	0
64	1	1	1	1	1	1

Jede Zeile in dieser Tabelle entspricht also gerade einer Fehlerkombination, beispielsweise die Fehlerkombination #4 einer Kombination der folgenden Fehlersignale:

Fault 1 = 0 (also nicht aktiv)	Fehleradresse und Einstellwert: 0000001 0
Fault 2 = 0 (also nicht aktiv)	Fehleradresse und Einstellwert: 0000010 0
Fault 3 = 0 (also nicht aktiv)	Fehleradresse und Einstellwert: 0000011 0
Fault 4 = 0 (also nicht aktiv)	Fehleradresse und Einstellwert: 0000100 0
Fault 5 = 1 (also aktiv)	Fehleradresse und Einstellwert: 0000101 1
Fault 6 = 1 (also aktiv)	Fehleradresse und Einstellwert: 0000111 1

Um jetzt genau diese eine Fehlerkombination einzustellen, stellt die Software nacheinander die Fehleradressen (vgl. Abb. 3.6) der obigen sechs Fehler nacheinander ein und setzt jeweils auf dem 8. Kanal den entsprechenden einzustellenden Wert („1“ zum Aktivieren bzw. „0“ zum Deaktivieren). Nach dem Setzen eines einzelnen Fehlersignals wartet das Programm die ausgewählte Wartezeit (s. o.) ab (damit die Fehler sicher gesetzt sind und das TXS-System genügend Zeit für die Verarbeitung hatte) und wechselt dann zum nächsten einzustellenden Fehlersignal. Nachdem alle Fehler einer bestimmten Fehlerkombination so eingestellt wurden, werden die Werte der zur Aufzeichnung ausgewählten SDO-Kanäle eingelesen und das Ergebnis danach in einer CSV-Datei tabellarisch abgelegt. Anschließend wird mit der nachfolgenden Fehlerkombination fortgefahren (usw.).

Nach Abschluss der Gesamtanalyse liegt im Ausführungsverzeichnis des Programms eine Datei mit dem Namen „result.csv“. Tab. 3.2 zeigt, wie der Inhalt dieser Datei für die Einstellungen in Abb. 3.9 aussehen könnte. Für die Erstellung dieses Beispiels wurden die Fehlersignale direkt mit den ersten sechs SDO-Ausgangskanälen verbunden, weswegen die Ergebniskombinationen (SDO Kanäle 1 – 6) exakt den Fehlerkombinationen (Fault 1 – 6) entsprechen (wodurch auch gezeigt wird, dass auf die hier vorgestellte Weise tatsächlich die vorgegebenen Fehlerkombinationen eingestellt wurden).

Tab. 3.2 Beispiel-Ergebnistabelle für 6 Fehler und 6 aufgezeichnete Kanäle

Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6	SDO Kanal 1	SDO Kanal 2	SDO Kanal 3	SDO Kanal 4	SDO Kanal 5	SDO Kanal 6
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	1	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0	1	0	1
0	0	0	1	1	0	0	0	0	1	1	0
0	0	0	1	1	1	0	0	0	1	1	1
0	0	1	0	0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0	1	0	0	1
0	0	1	0	1	0	0	0	1	0	1	0
0	0	1	0	1	1	0	0	1	0	1	1
0	0	1	1	0	0	0	0	1	0	0	0
0	0	1	1	0	1	0	0	1	1	0	1
0	0	1	1	1	0	0	0	1	1	1	0
0	0	1	1	1	1	0	0	1	1	1	1
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	0	1	0	0	1	0	0	1	0
0	1	0	0	1	1	0	1	0	0	1	1
0	1	0	1	0	0	0	1	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1	0	1	1	0
0	1	0	1	1	1	0	1	0	1	1	1
0	1	1	0	0	0	0	1	1	0	0	0
0	1	1	0	0	1	0	1	1	0	0	1
0	1	1	0	1	0	0	1	1	0	1	0
0	1	1	0	1	1	0	1	1	0	1	1
0	1	1	1	0	0	0	1	1	1	0	0
0	1	1	1	0	1	0	1	1	1	0	1
0	1	1	1	1	0	0	1	1	1	1	0
0	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0
1	0	0	0	1	1	1	0	0	0	1	1
1	0	0	1	0	0	1	0	0	1	0	0
1	0	0	1	0	1	1	0	0	1	0	1
1	0	0	1	1	0	1	0	0	1	1	0
1	0	0	1	1	1	1	0	0	1	1	1

Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6	SDO Kanal 1	SDO Kanal 2	SDO Kanal 3	SDO Kanal 4	SDO Kanal 5	SDO Kanal 6
1	0	1	0	0	0	1	0	1	0	0	0
1	0	1	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0
1	0	1	0	1	1	1	0	1	0	1	1
1	0	1	1	0	0	1	0	1	1	0	0
1	0	1	1	0	1	1	0	1	0	0	1
1	0	1	1	1	0	1	0	1	1	1	0
1	0	1	1	1	1	1	0	1	1	1	1
1	1	0	0	0	0	1	1	0	0	0	0
1	1	0	0	0	1	1	1	0	0	0	1
1	1	0	0	1	0	1	1	0	0	1	0
1	1	0	0	1	1	1	1	0	0	1	1
1	1	0	1	0	0	1	1	0	1	0	0
1	1	0	1	0	1	1	1	0	0	0	1
1	1	0	1	1	0	1	1	0	1	1	0
1	1	0	1	1	1	1	1	0	1	1	1
1	1	1	0	0	0	1	1	1	0	0	0
1	1	1	0	0	1	1	1	1	0	0	1
1	1	1	0	1	0	1	1	1	0	1	0
1	1	1	0	1	1	1	1	1	0	1	1
1	1	1	1	0	0	1	1	1	1	0	0
1	1	1	1	0	1	1	1	1	0	0	1
1	1	1	1	1	0	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1

Als Beispiel für die Anwendung der in den vorangegangen Abschnitten beschriebenen Methode wurde das Verhalten eines UND-Funktionsblocks (mit zwei Eingängen) untersucht. Hierfür wurde der in Abb. 3.2 dargestellte Funktionsplan verwendet. In diesem Funktionsplan sind vier Fehlersignale definiert, die folgende Fehler steuern:

Fehler 1 – NSF des ersten Eingangssignals („Digitaler Eingang 1“)
Fehler 2 – NSF des zweiten Eingangssignals („Digitaler Eingang 2“)
Fehler 3 – SF des ersten Eingangssignals („Digitaler Eingang 1“)
Fehler 4 – SF des zweiten Eingangssignals („Digitaler Eingang 2“)

Aufgezeichnet wurden die folgenden Signale:

Ausgang UND (SDO Kanal 1)
Fehlerattribut UND (SDO Kanal 2)

Die für die Analyse vorgenommenen Einstellungen entsprachen somit den in Abb. 3.10 dargestellten Werten.

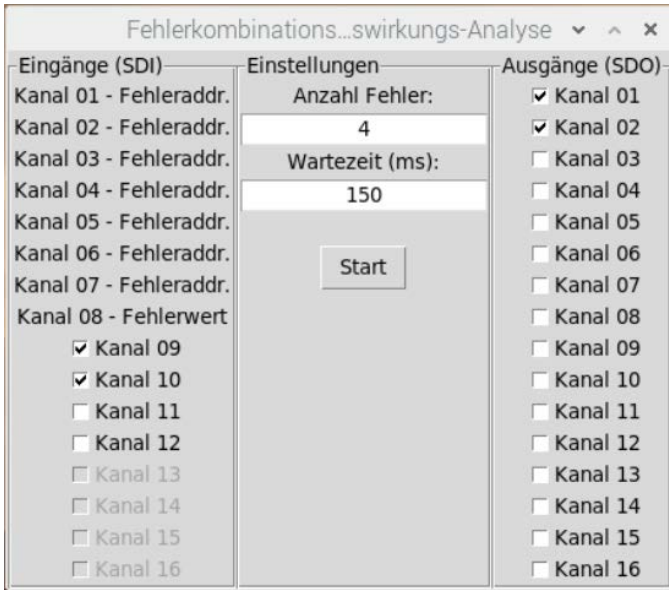


Abb. 3.8 Im Programm „Fehlerkombinationsauswirkungs-Analyse“ vorgenommene Einstellungen für die im Text beschriebene Beispielanwendung.

Die Ergebnistabelle der entsprechenden Analyse ist in Tab. 3.3 wiedergegeben. Für eine bessere Lesbarkeit wurden in dieser Tabelle die Spaltenüberschriften gemäß obiger Beschreibung angepasst.

Tab. 3.3 Ergebnis der automatischen Analyse

NSF Signal 1	NSF Signal 2	SF Signal 1	SF Signal 2	Ausgang UND	Fehlerattr. UND
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

Insgesamt wurden gemäß dieser Tabelle in der automatischen Analyse 16 unterschiedliche Fehlerkombinationen eingestellt und für jede dieser Kombinationen der Ausgangswert des UND-Funktionsblocks sowie dessen Fehlerattribut aufgezeichnet. Die aufgezeichneten Werte für den Ausgang und das Fehlerattribut des Ausgangs des UND-Funktionsblocks entsprechen der Erwartung (aus der Beschreibung dieses Funktionsblocks im TXS-Handbuch /TXS 12/).

Anmerkung:
In diesem einfachen Beispiel wurden für die Analyse auch unnötige Fehlerkombinationen eingestellt. Beispielsweise sind in der 6. Fehlerkombination in Tab. 3.3 gleichzeitig der nicht-selbstmeldende Fehler (NSF) und der selbstmeldende Fehler (SF) des zweiten Signals berücksichtigt. Dies entspricht in der Wirkung auf das Signal 2 genau der zweiten Fehlerkombination, in der nur der selbstmeldende Fehler (SF) von Signal 2 aktiviert ist (vgl. hierzu auch deren Verknüpfung in Abb. 3.2). Hier könnte zukünftig im Sinne einer Optimierung die Software derart verändert werden, dass solche unnötigen Fehlerkombinationen übersprungen werden (und somit die Analyse schneller beendet werden kann).

3.3 Monte-Carlo-Simulationen im AnTeS

Anstatt, wie im Abschnitt 3.2 beschrieben, systematisch alle Fehlerkombination nacheinander einzustellen und deren jeweilige Gesamtauswirkung aufzuzeichnen (automatisierte FMEAs), können ansteuerbare Fehler auch zufällig gemäß ihrer statistischen Wahrscheinlichkeit aktiviert werden. Auf diese Weise lassen sich Modellsysteme im Modul 2 des AnTeS auch für Monte-Carlo-Simulationen verwenden, wodurch unmittelbar auch statistische Zuverlässigkeitskenndaten ermittelbar sind.

Die statistische Ansteuerung von Fehlern kann im Modul 2 innerhalb von Simulink selbst erfolgen. Eine ausführlichere Beschreibung hierzu befindet sich im Abschnitt 4.2 („Sensitivitätsanalysen mit AnTeS“) im Rahmen der durchgeführten Tests mit AnTeS für ausgewählte Modellsysteme.

4 Durchgeführte Analysen und Tests

Im vorangegangenen Kapitel wurden neu entwickelte Methoden und Werkzeuge für die Untersuchung digitaler Leittechniksysteme vorgestellt. In diesem Kapitel wird anhand konkret durchgeführter Analysen demonstriert, wie diese neuen Werkzeuge für verschiedene Fragestellungen eingesetzt werden können. Darüber hinaus stellt dies auch einen Test der im Vorgängervorhaben entwickelten Methoden zur Sensitivitätsanalyse dar, was ein wichtiges Ziel des aktuellen Vorhabens war.

4.1 Automatisierte Fehlerauswirkungsanalysen mit AnTeS

Mit Hilfe der neu entwickelten Methoden und Werkzeuge wurden u. a. ausgewählte Modellsysteme aus dem Vorgängervorhaben untersucht. Eine detaillierte Beschreibung dieser Modellsysteme findet sich in /PIL 18/, für das bessere Verständnis an dieser Stelle ist der entsprechende Modellierungsansatz zusätzlich auch im Anhang (siehe Abschnitt A.3) dieses Dokuments nachzulesen.

Das Modellsystem A222 besteht nur aus Komponenten einer einzelnen Leittechnikplattform („A“) und setzt sich aus 2 Voting Units (VU), 2 Processing Units (PU) und 2 Acquisition Units (AU) zusammen (vgl. Abb. 4.1).

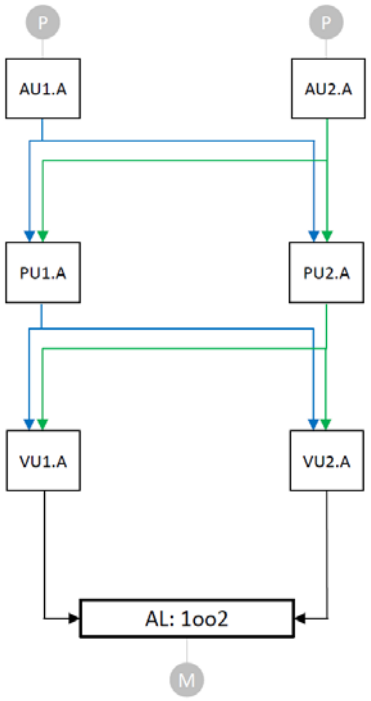


Abb. 4.1 Architektur des Modellsystems A222 (aus /PIL 18/).

Das Modellsystem A222 wurde sowohl mit dem Modul 1 als auch dem Modul 2 von AnTeS mit allen in /PIL 18/ beschriebenen Funktionalitäten reproduziert (vgl. Abb. 4.2 und Abb. 4.3). Insbesondere im Modul 2 (vgl. Abb. 4.3) lässt sich die Struktur von A222 gut erkennen. Auf der linken Seite in Abb. 4.3 befinden sich 20 Eingänge, über diese kann das Modellsystem nach der Umwandlung in eine DLL (siehe hierzu Anhang A.2) von außen angesteuert werden („Externe Ansteuerung“). Hierzu gehört einerseits die Ansteuerung der zu- und wegschaltbaren Fehler/Ausfälle (über die Speicher rechts oben in der Abbildung), als auch das Starten von Reparaturen und wiederkehrenden Prüfungen („Reparaturen und WKPen“). Die Berücksichtigung von Reparaturen und WKPen spielt für automatische Fehlerauswirkungsanalysen keine Rolle. Diese sind allerdings bei der Durchführung von Monte-Carlo-Simulationen wichtig und werden im nachfolgenden Abschnitt 4.2 näher beleuchtet.

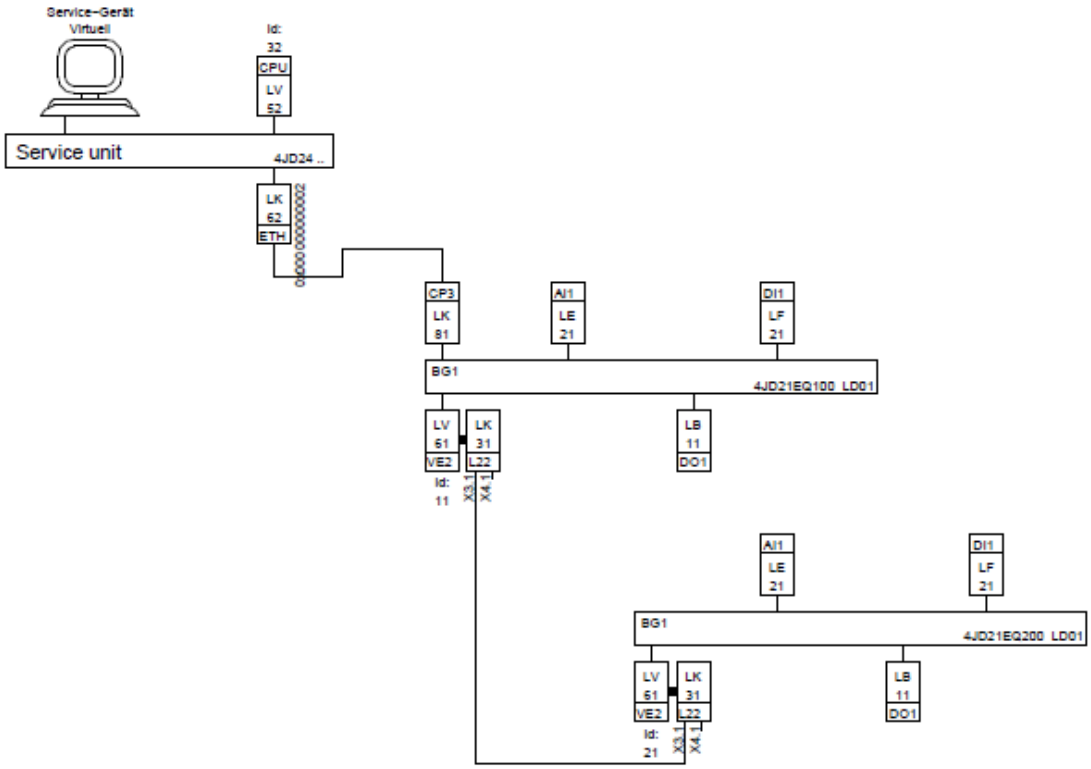


Abb. 4.2 Übersicht (Netzwerkplan) des Modellsystems A222 im Modul 1 (TXS) des AnTeS

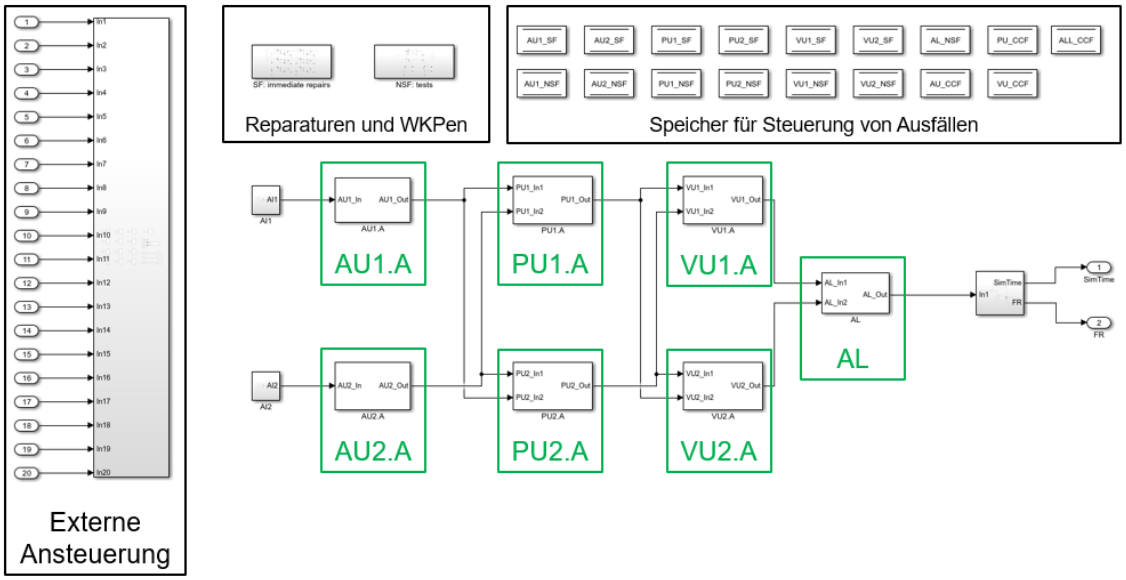


Abb. 4.3 Übersicht des Modellsystems A222 im Modul 2 (Simulink) des AnTeS.

Da im Modul 2 des AnTeS dieselben Funktionsblöcke wie im Modul 1 des AnTeS zur Verfügung stehen (siehe Abschnitt 2.2), können sowohl die leittechnischen Funktionen als auch die Ansteuerung von Fehlern/Ausfällen in beiden Modulen auf dieselbe Weise realisiert werden. In beiden Fällen wird jeweils das zweite Maximum der beiden Eingangswerte mit einem Grenzwert (im Grenzwertbaustein – „comp“) verglichen und ggfs. ein Auslösesignal an die nachfolgenden Voting Units weitergeleitet. Dieses Signal kann dabei zusätzlich auch noch per Fehlerinjektion mit einem selbstmeldenden (SF) oder nicht-selbstmeldenden (NSF) Fehler versehen werden („Ansteuerung von Fehlern/Ausfällen“ in Abb. 4.4 und Abb. 4.5, vgl. hierzu Abschnitt 3.1).

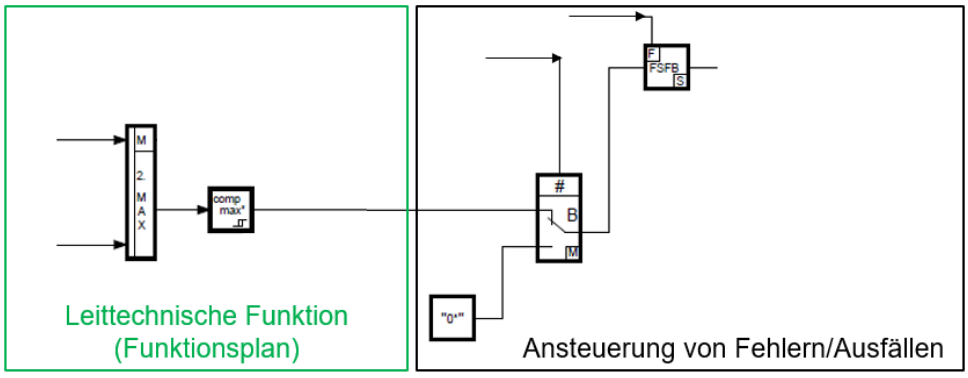


Abb. 4.4 Leittechnische Funktion (Funktionsplan) in der Processing Unit 1 (PU1.A) des Modellsystems A222 im Modul 1 von AnTeS und Ansteuerung von Fehlern/Ausfällen.

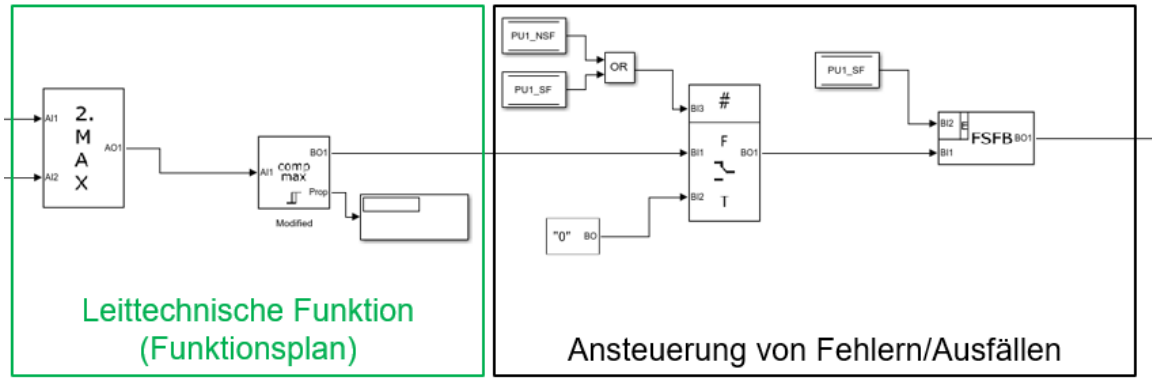


Abb. 4.5 Leittechnische Funktion (Funktionsplan) in der Processing Unit 1

Hier: PU1.A des Modellsystems A222 im Modul 2 von AnTeS und Ansteuerung von Fehlern/Ausfällen

Mit beiden Modulen von AnTeS wurden automatisierte Fehlerauswirkungsanalysen durchgeführt (vgl. Abschnitt 3.2). Für das Modellsystem A222 liefert dabei sowohl das Modul 1 als auch das Modul 2 des AnTeS dieselben Ergebnisse. In Tab. 4.1 sind die mit dem Modul 2 des AnTeS gewonnenen sogenannten Minimalschnitte (Ausfallkombinationen von Fehlern) von A222 aufgelistet. Jede Zeile der Tabelle stellt dabei eine Kombination von (für dieses Modellsystem ein oder maximal zwei) Ausfällen dar, durch die A222 insgesamt als ausgefallen angenommen werden muss. Zum Vergleich zeigt Abb. 4.6 die Ergebnisse für das Modellsystem A222 aus dem Vorgängervorhaben /PIL 18/, diese wurden mit der PSA-Software RiskSpectrum /RIS 21/ gewonnen.

Tab. 4.1 Minimalschnitte für das Modellsystem A222 im Modul 2 des AnTeS (Modul 1 identisch)

Event 1	Event 2
CCF ALL	
CCF VU	
CCF PU	
CCF AU	
AL NSF	
AU2.A NSF	
AU1.A NSF	
VU1.A NSF	VU2.A NSF
VU2.A SF	VU1.A NSF
VU1.A SF	VU2.A NSF
VU1.A SF	VU2.A SF
PU1.A NSF	PU2.A NSF
PU2.A SF	PU1.A NSF
PU1.A SF	PU2.A NSF
PU1.A SF	PU2.A SF
AU1.A SF	AU2.A SF

CCF – Common Cause Failure
ALL – VU und PU und AU
VU – Voting Unit
PU – Processing Unit
AU – Acquisition Unit
NSF – Nicht-selbstmeldender Fehler
SF – Selbstmeldender Fehler

No	Probability	%	Event 1	Event 2
1	5,60E-05	48,66	AU1.A NSF	
2	5,60E-05	48,66	AU2.A NSF	
3	7,46E-07	00,65	CCF VU	
4	7,46E-07	00,65	CCF ALL	
5	7,46E-07	00,65	CCF PU	
6	7,46E-07	00,65	CCF AU	
7	3,44E-08	00,03	AL NSF	
8	2,82E-08	00,02	AU1.A SF	AU2.A SF
9	1,58E-08	00,01	PU1.A SF	PU2.A SF
10	7,03E-09	00,01	PU1.A NSF	PU2.A SF
11	7,03E-09	00,01	PU1.A SF	PU2.A NSF
12	3,14E-09	00,00	PU1.A NSF	PU2.A NSF
13	3,14E-09	00,00	VU1.A NSF	VU2.A NSF
14	3,12E-09	00,00	VU1.A NSF	VU2.A SF
15	3,12E-09	00,00	VU1.A SF	VU2.A NSF
16	3,11E-09	00,00	VU1.A SF	VU2.A SF

Abb. 4.6 Minimalschnitte für das Modellsystem A222 in RiskSpectrum.

Die mit dem AnTeS gewonnenen Ergebnisse stimmen mit den Ergebnissen aus dem Vorgängervorhaben /PIL 18/ überein. Unterschiede gibt es lediglich hinsichtlich der Reihenfolge der Minimalschnitte. Während in AnTeS die gefundenen Minimalschnitte gemäß der Reihenfolge auftauchen in der diese getestet wurden, werden diese für das Modellsystem in RiskSpectrum gemäß ihrer Eintrittswahrscheinlichkeit sortiert. Wie im nachfolgenden Abschnitt dargelegt, können mit dem AnTeS auch Monte-Carlo-Simulationen durchgeführt werden. Diese liefern dann auch mit dem AnTeS die Wahrscheinlichkeiten, mit denen die einzelnen Ausfallkombinationen auftreten. Insgesamt liefert das AnTeS somit dieselben Ergebnisse wie eine klassische Analyse mit Hilfe von Fehlerbäumen mit Hilfe der Software RiskSpectrum.

Grundsätzliche Unterschiede bei den verschiedenen Vorgehensweisen bestehen jedoch hinsichtlich des zeitlichen Aufwands und der durchzuführenden Arbeiten. Während die Analyse mit RiskSpectrum die händische Erstellung von Fehlerbäumen (mit ggfs. vorausgehender FMEA) erfordert, werden bei Verwendung des AnTeS Abbilder des zu untersuchenden Systems erstellt und anschließend automatisch analysiert.

Bei Verwendung vom Modul 2 des AnTeS erfordert die eigentliche Analyse (nach Erstellung des Modells) weniger als eine Sekunde (für 131.072 zu testende Fehlerkombinationen im Fall des Modellsystems A222), da Berechnungen im Modul 2 des AnTeS wesentlich schneller als in Echtzeit durchgeführt werden können. Bei Verwendung des Moduls 1 des AnTeS erhöht sich die benötigte Analysezeit für eine automatische Fehlerauswirkungsanalyse (nach Erstellung des Modells) für das Modellsystem A222 derzeit noch auf ca. 6 Tage 10 Stunden 22 Minuten (getestet) bei Verwendung der Fehleradressierung, wie sie in Abschnitt 3.1 beschrieben wird.

Zukünftig kann die Durchführung automatisierter Fehlerauswirkungsanalysen mit Hilfe des Moduls 1 des AnTeS allerdings noch deutlich beschleunigt werden. Erfolgt die Ansteuerung der Fehler nämlich direkt über die Netzwerkanbindung des Servicerechners (siehe hierzu Anhang A.1), so können Ausfallkombinationen wesentlich schneller im Modul 1 des AnTeS eingestellt werden. In diesem Fall wäre die Geschwindigkeit nur noch durch die (typischen) 50-ms-Zyklen des TXS begrenzt, d. h. die Neuberechnungen des Gesamtzustandes alle 50 ms. Für eine sicher fehlerfreie Analyse sollten für jede Fehlerkombination, die getestet werden soll, dann nur noch 2 Zyklen (100 ms) benötigt werden, was für 131.072 Kombinationen (A222) rechnerisch dann insgesamt weniger als vier Stunden dauern würde. Die Analysezeit kann, wo möglich, durch Reduzierung der TXS-Zykluszeiten noch weiter reduziert werden.

Beispielsweise würde für eine Zykluszeit von 16 ms (was der Einstellung des ursprünglichen Projekts auf den Rechnern des TXS – Turbinensteuerung KKK – bei der Auslieferung an die GRS entspräche) dann rechnerisch deutlich weniger als eine Stunde benötigt.

Vergleichbare Analysen wurden auch für die Modellsysteme A133 und A222mod (siehe /PIL 18/) durchgeführt, aufgrund des sehr hohen Zeitaufwands bei der Verwendung des Moduls 1 des AnTeS jedoch ausschließlich mit dem Modul 2 des AnTeS. Auch für diese Modellsysteme lieferte AnTeS identische Ergebnisse wie im Vorgängervorhaben.

4.2 Sensitivitätsanalysen mit AnTeS

Während für automatisierte Fehlerauswirkungsanalysen Fehlerkombinationen in Modellsystemen direkt von außen vorgegeben werden, müssen diese in Monte-Carlo-Simulationen statistisch gemäß ihrer Eintrittswahrscheinlichkeit quasi „von selbst“ auftreten.

Daher wurden für die in diesem Abschnitt beschriebenen Sensitivitätsanalysen, anstatt binärer Steuersignale (True/False), die Eintrittswahrscheinlichkeiten von Fehlern an das jeweilige Modellsystem übergeben. Abb. 4.7 zeigt dies repräsentativ für das Modellsystem A222 (eine Übersicht des Modellsystems ist in Abb. 4.3 dargestellt).

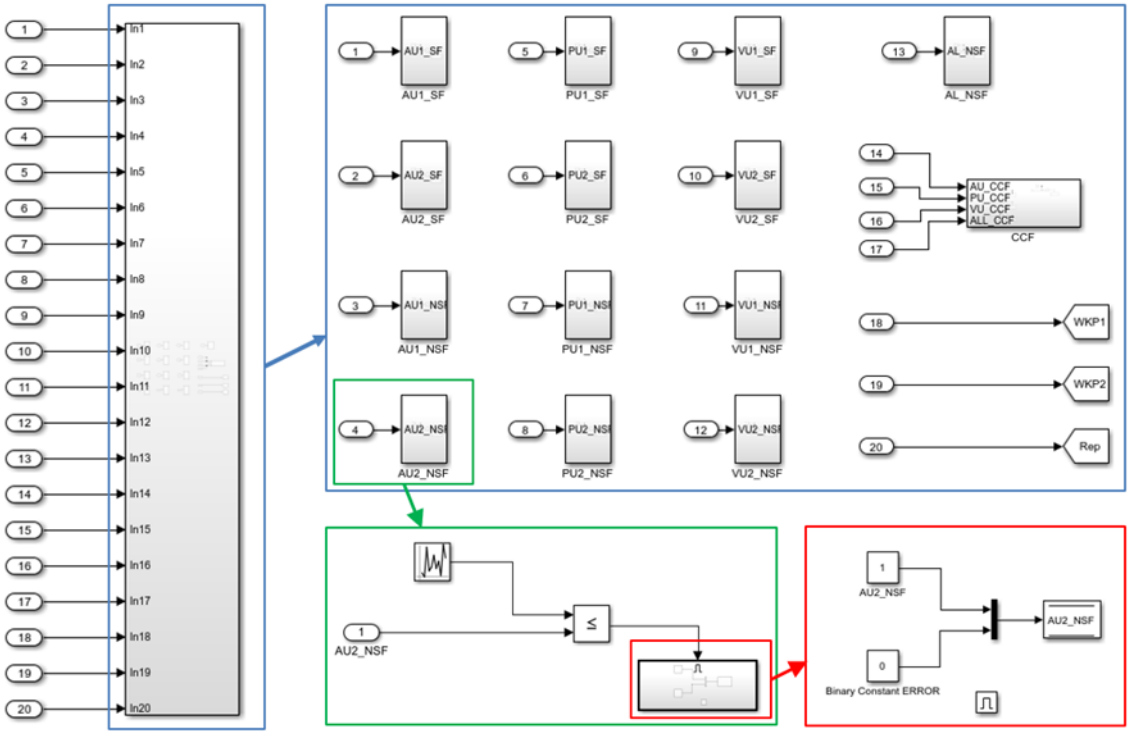


Abb. 4.7 Ansteuerung des Modellsystems A222 von außen im Modul 2 des AnTeS bei Monte-Carlo-Simulationen.

Die an das Modellsystem übergebenen Wahrscheinlichkeiten (in Form von Vergleichswerten für den Vergleich mit Zufallszahlen, Eingänge 1 bis 17) werden in ein Simulink-Subsystem (linker blauer Kasten in Abb. 4.7) eingelesen. Dieses Subsystem setzt sich aus einer Reihe weiterer Subsysteme zusammen (rechter blauer Kasten in Abb. 4.7). In diesen Subsystemen werden die Vergleichswerte („Wahrscheinlichkeiten“) bei jedem Berechnungsschritt mit intern erzeugten (Pseudo-)Zufallszahlen verglichen (in Abb. 4.7 exemplarisch für den nicht-selbstmeldenden Ausfall AU2_NSF der AU 2 im grünen Kasten dargestellt). Ist die Zufallszahl kleiner als der von außen vorgegebene Vergleichswert, so liefert der Vergleichsbaustein („≤“) eine logische „1“, die ein getriggertes Subsystem startet. Dieses setzt schließlich den entsprechenden Fehler (roter Kasten in Abb. 4.7).

Die an das Modell übergebenen Vergleichswerte repräsentieren dabei die Ausfallwahrscheinlichkeiten in dem Zeitraum, welcher zwischen zwei Berechnungsschritten im Modell vergeht. In allen hier vorgestellten Analysen wurde als Zeitabstand zwischen zwei Berechnungen (Zeitschritte in der Simulation) jeweils eine Stunde angenommen. Dadurch können direkt die im Vorgängervorhaben verwendeten Werte (in h^{-1}) benutzt

werden, allerdings zusätzlich multipliziert mit dem Wertebereich der verwendeten Zufallszahlen.

Dieser Zusammenhang wird durch das Beispiel in Abb. 4.8 verdeutlicht. Die darin blau markierten Simulink-Funktionsblöcke erzeugen bei jedem Berechnungsschritt eine Zufallszahl (in einem vorgegebenen Werte-Intervall, konkrete Werte siehe Abbildung), diese werden mit Vergleichswerten (rot umrahmt) verglichen.

Zeichnet man in jedem Zeitschritt die aktuellen Fehlerraten (FR1, …, FR5) auf und stellt diese als Funktion der (Simulations-)Zeit dar, so ergibt sich das Diagramm in Abb. 4.9. Wie gewünscht und erwartet, pendeln sich die Fehlerraten bei einem Wert von $1 \cdot 10^{-7}$ h^{-1} ein (in diesem konkreten Fall ab etwa 200.000 Jahren vergangener (Simulations-)Zeit).

Die erwarteten (und in diesem einfachen Beispiel gleichen) Fehlerraten lassen sich aus dem Verhältnis des jeweiligen Vergleichswerts zu maximalem Wert der entsprechenden Zufallszahl bestimmen:

$$FR1, \dots, FR5 = \frac{1 \cdot 10^3}{1 \cdot 10^{10}} = \frac{1 \cdot 10^4}{1 \cdot 10^{11}} = \frac{1 \cdot 10^5}{1 \cdot 10^{12}} = \frac{1 \cdot 10^6}{1 \cdot 10^{13}} = \frac{1 \cdot 10^7}{1 \cdot 10^{14}} = 1 \cdot 10^{-7}\ [h^{-1}]$$

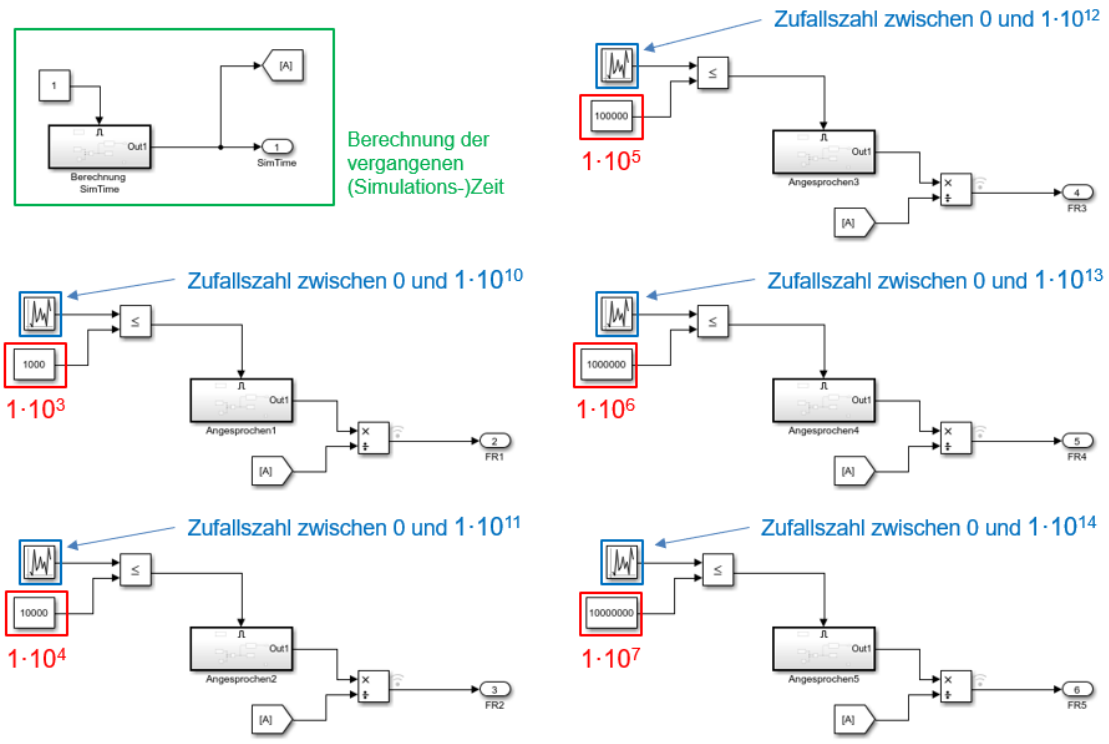


Abb. 4.8 Beispielhafte Generierung von Fehlerraten im Modul 2 des AnTeS (in diesem Beispiel der Fehlerraten FR1, …, FR5 mit jeweils $1 \cdot 10^{-7}$ h^{-1}).

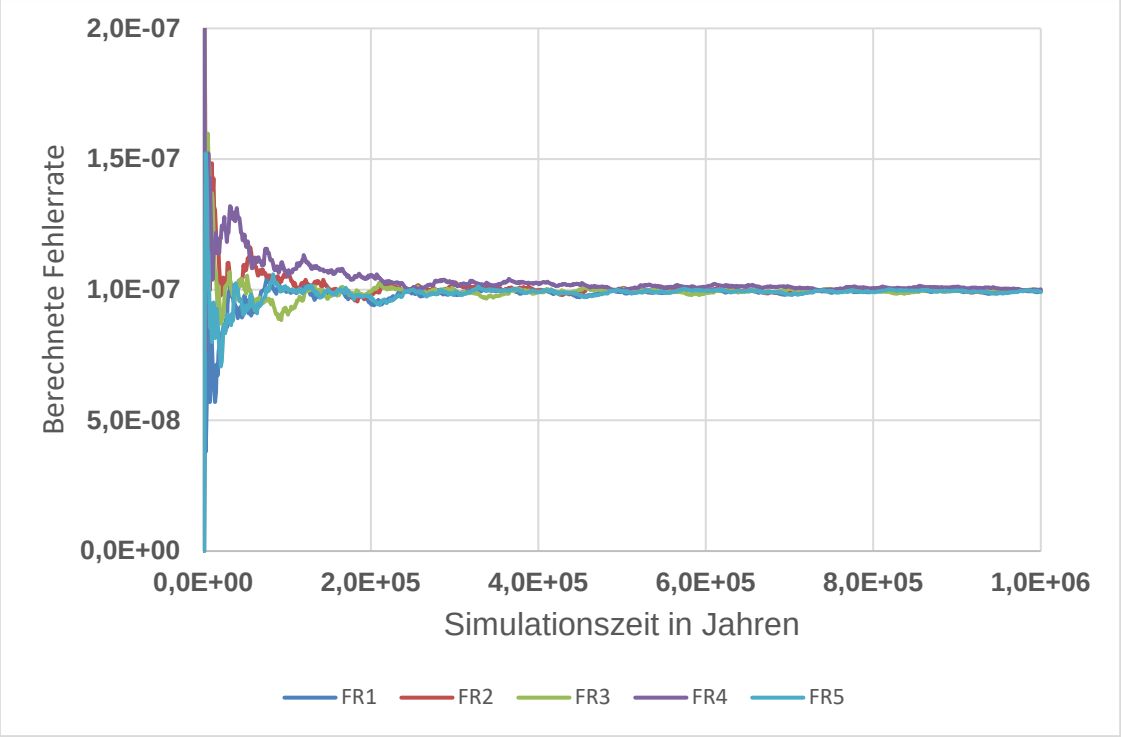


Abb. 4.9 Für das Beispiel in Abb. 4.8 berechnete Fehlerraten (FR1, …, FR5) als Funktion der Simulationszeit.

Wie bei der Modellierung der Modellsysteme im Vorgängervorhaben /PIL 18/ mit RiskSpectrum /RIS 21/, müssen auch bei den Simulationen mit dem Modul 2 des AnTeS zusätzlich noch unmittelbare Reparaturen (von selbstmeldenden Fehlern (SF)) und Reparaturen nach WKPen berücksichtigt werden, welche gesetzte Fehler nach einer definierten Reparaturzeit wieder „löschen“. Hierzu dienen entsprechende Subsysteme (vgl. Abb. 4.10 und Abb. 4.12). Die Reparaturzeit für beide Ausfallarten (SF und NSF) wird dem Modellsystem von außen als Zahlenwert übergeben (Signal Nr. 20 („Rep“) im blauen Kasten in Abb. 4.7) und kann daher für Sensitivitätsanalysen flexibel variiert werden. Ebenso werden die zeitlichen Abstände zwischen WKPen für die beiden Redundanzen flexibel änderbar an das Modellsystem übergeben (Signale 18 („WKP1“) und 19 („WKP2“) im blauen Kasten in Abb. 4.7).

4.2.1 Berücksichtigung von sofortigen Reparaturen

Abb. 4.10 zeigt die Berücksichtigung sofortiger Reparaturmaßnahmen (mit einer vorgegebenen Reparaturzeit) bei selbstmeldenden Fehlern innerhalb des Modellsystems A222 in einem eigenen Simulink-Subsystem (vgl. auch Abb. 4.3).

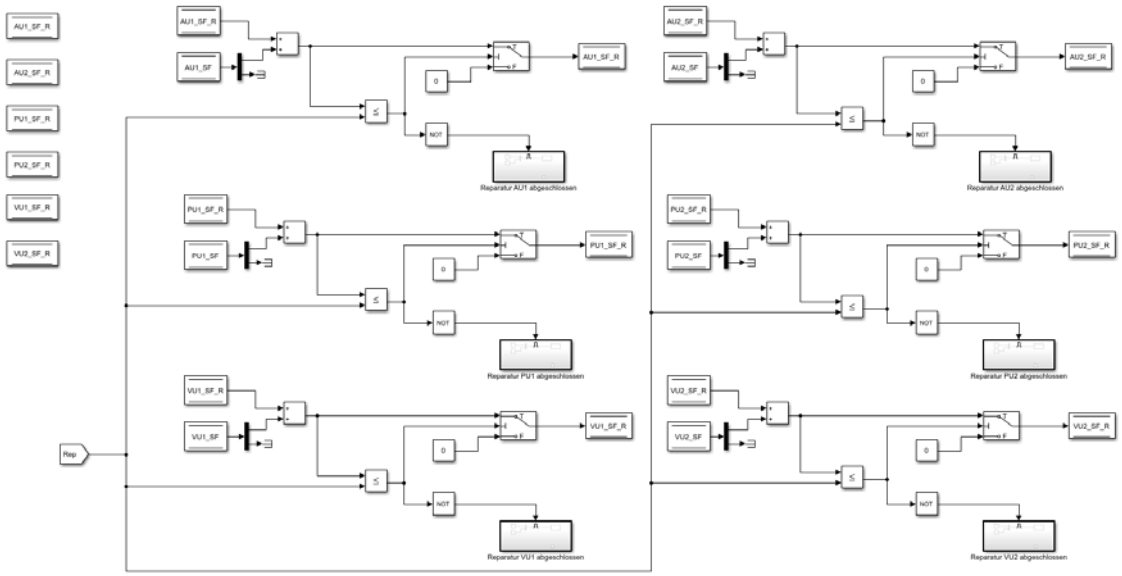


Abb. 4.10 Simulink-Subsystem, dass im Modellsystem A222 sofortige Reparaturen durchführt.

Die zugrundeliegende Idee, wie (automatische) sofortige Reparaturen aufgetretener selbstmeldender Fehler berücksichtigt werden, kann exemplarisch für selbstmeldende Fehler (SF) der Voting Unit 1 (VU 1) in Abb. 4.11 nachvollzogen werden.

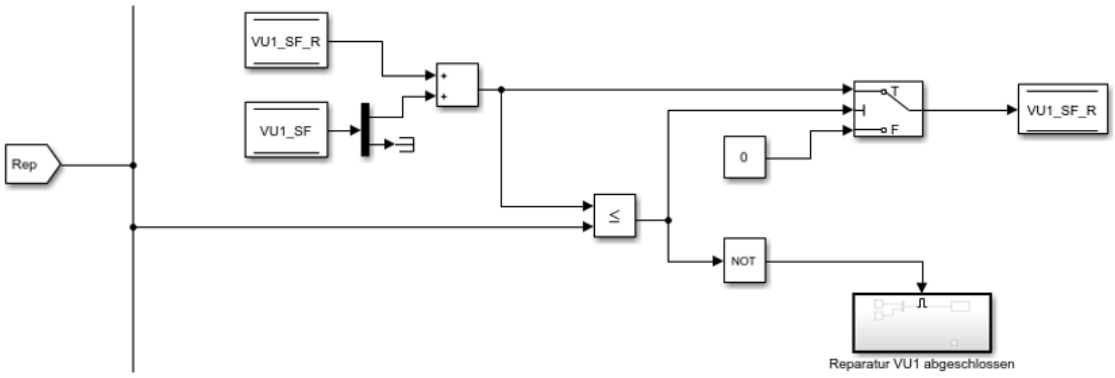


Abb. 4.11 Sofortige Reparaturen von selbstmeldenden Fehlern (SF) der VU1 (Detailansicht aus Abb. 4.10).

Hierbei wird exemplarisch angenommen, dass die ans Modellsystem übergebene Reparaturzeit 8 h beträgt („Rep“ auf der linken Seite in Abb. 4.11) und die Zeitschritte zwischen zwei Berechnungen im Modellsystem 1 h entsprechen.

Zu Beginn soll noch kein selbstmeldender Fehler vorliegen (der Wert im Speicher „VU1_SF“ also „0“ sein, d. h. der Fehler wurde noch nicht gesetzt). Zusätzlich soll im Speicher „VU1_SF_R“ gerade ein Wert von „0“ liegen (dieser Wert wird nachfolgend nachvollziehbar). In diesem Fall ist der aufsummierte Wert (quadratisches Symbol mit zwei Pluszeichen) von „VU1_SF_R“ und „VU1_SF“ auch gleich null („0 + 0“). Der Vergleich dieser Summe mit der vorgegebenen Reparaturzeit (Funktionsblock „≤“) liefert dann am Ausgang eine 1 (True). Dadurch leitet der ebenfalls im Bild sichtbare Umschalter den Wert am oberen Eingang weiter (im Baustein mit „T“ für „True“ gekennzeichnet), also ebenfalls eine 0. Somit wird im selben Berechnungsschritt auch wieder in den Speicher „VU_SF_R“ eine 0 geschrieben. Das Subsystem rechts unten im Bild („Reparatur VU1 abgeschlossen“) erhält aufgrund der Negierung („NOT“) in diesem Fall ebenfalls kein Triggersignal, es finden dann auch keine Berechnungen in diesem Subsystem statt.

Tritt jetzt ein selbstmeldender Fehler in der VU 1 auf (wird also der Speicher „VU1_SF“ auf „1“ gesetzt), ändert sich das Verhalten wie folgt: Die Summe aus „VU_SF_R“ und „VU_SF“ wird 1 („0 + 1“), was aber noch immer kleiner als die Reparaturzeit „Rep“ ist. Dementsprechend wird immer noch das obere Signal am Eingang vom Umschalter weitergeleitet, dieses beträgt jetzt jedoch 1. Somit wird auch eine 1 in den Speicher „VU1_SF_R“ geschrieben, dieser hat sich also um 1 erhöht.

Mit jedem Berechnungsschritt erhöht sich der in „VU1_SF_R“ gespeicherte Wert jeweils um 1. Dies jedoch nur solange, bis die Summe aus „VU1_SF_R“ und „VU_SF“ nicht mehr kleiner oder gleich der Reparaturzeit ist. Dies geschieht nach acht Berechnungsschritten (entsprechend also 8 h simulierter Zeit). Dann gibt der Vergleicher (Symbol „≤“) eine logische 0 (False) aus. Dadurch schaltet der Umschalter auf den unteren Eingang (gekennzeichnet mit „F“ – „False“) um, an welchem eine konstante 0 ansteht. Hierdurch wird in den Speicher „VU1_SF_R“ eine 0 geschrieben. Gleichzeitig wird aufgrund der Negierung („NOT“) das Subsystem („Reparatur VU1 abgeschlossen“) getriggert und dadurch auch der selbstmeldende Fehler (Speicher „VU1_SF“) gelöscht (der Fehler also „repariert“). Anschließend ist die Gesamtsituation wieder genauso wie vor dem Auftreten des selbstmeldenden Fehlers.

Auf dieselbe Weise werden alle selbstmeldenden Fehler, welche im Modellsystem A222 auftreten können, behandelt (siehe Abb. 4.10). Jeder selbstmeldende Fehler wird also jeweils genau nach der von außen vorgegebenen Reparaturzeit automatisch beseitigt.

4.2.2 Berücksichtigung von WKPen

Die Berücksichtigung von nicht-selbstmeldenden Fehlern bei WKPen ist im Vergleich zu den sofortigen Reparaturen von selbstmeldenden Fehlern vergleichsweise einfacher nachvollziehbar. Das entsprechende Subsystem des Modellsystems A222 ist in Abb. 4.12 dargestellt. In der oberen Hälfte wird im Wesentlichen ein „Timer“ (in einem Speicherbaustein) in jedem Zeitschritt hochgezählt und mit den von außen vorgegebenen Abständen zwischen WKPen in den einzelnen Redundanzen („WKP1“ und „WKP2“) verglichen. Nach den eingestellten Zeiten wird jeweils ein Subsystem für den Start der jeweiligen WKP getriggert und nach Beendigung beider WKPen wieder auf seinen Startwert zurückgesetzt[3]. Die getriggerten Subsysteme („Starte WKP Red. 1“ und „Starte WKP Red. 2“) in der oberen Hälfte von Abb. 4.12 setzen jeweils die Speicher „Rep_Red_1“ bzw. „Rep_Red_2“ auf 1.

In der unteren Hälfte der Abb. 4.12 werden nach dem Beginn der WKPen jeweils die entsprechenden Speicher „Rep_Red_1“ bzw. „Rep_Red_2“ jeweils um 1 erhöht (durch die Subsysteme „Erhöhe Rep_Red_1 um 1“ bzw. „Erhöhe Rep_Red_2 um 1“, wenn diese einen Wert > 0 haben, bis der Vergleich mit der Reparaturzeit („Rep“) die entsprechenden Subsysteme „WKP Red. 1 abgeschlossen“ bzw. „WKP Red. 2 abgeschlossen“ startet. Diese löschen sämtliche nicht-selbstmeldenden Fehler aller Komponenten der jeweiligen Redundanz und setzen auch die Speicher „Rep_Red_1“ bzw. „Rep_Red_2“ wieder auf 0.

[3] Bei dieser Art der Modellierung benötigt die Fehlersuche bei einer WKP (unwesentlich vereinfachend) keine Zeit, gefundene Fehler werden dann allerdings nicht sofort gelöscht. Vielmehr werden gefundene (zuvor unerkannte) Fehler erst nach einer definierten Reparaturzeit entfernt (siehe nachfolgenden Abschnitt im Text). Außerdem fanden in den in diesem Vorhaben betrachteten Fällen die WKPen abwechselnd in unterschiedlichen Redundanzen statt.

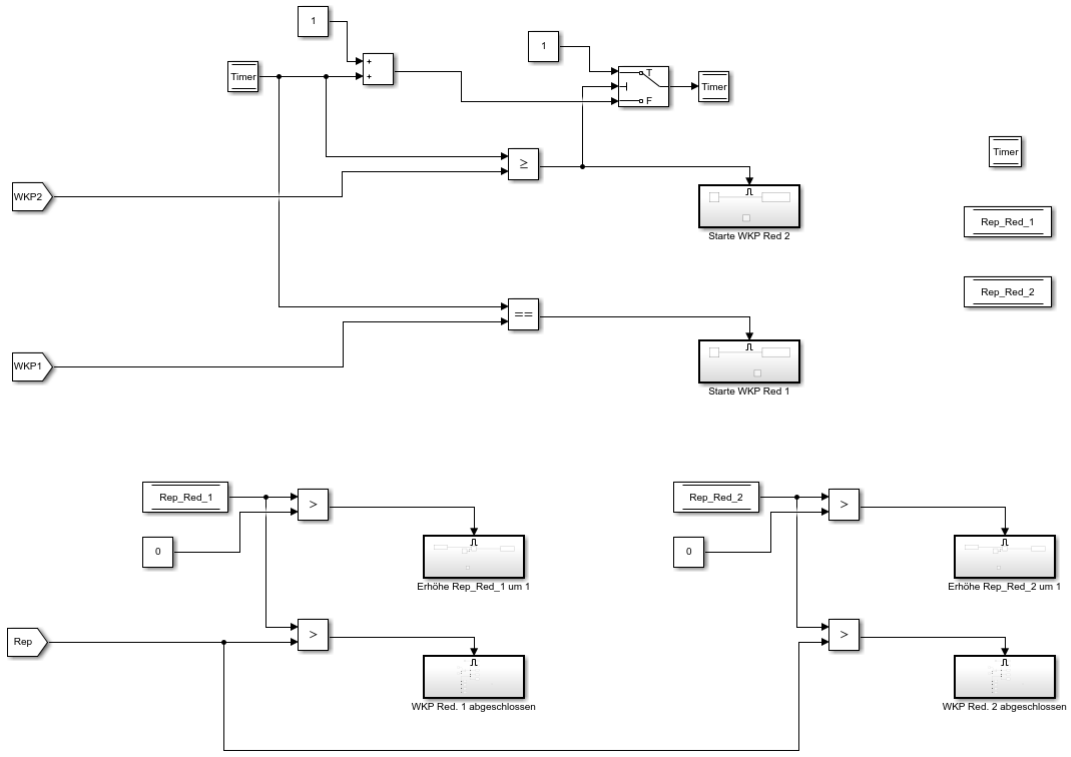


Abb. 4.12 Simulink-Subsystem zur Durchführung von WKPen im Modellsystem A222

Insgesamt werden durch das Subsystem in Abb. 4.12 also jeweils alle nicht-selbstmeldenden Fehler für jede Redundanz einzeln nach der jeweiligen Zeit, die zwischen zwei WKPen in diesen Redundanzen vergeht (zuzüglich der eingestellten Reparaturzeit), automatisch gelöscht. Dies entspricht dem angestrebten Verhalten des Modellsystems A222, dass nach einer WKP in einer Redundanz jeweils alle fehlerhaften Komponenten (mit NSF) repariert werden.

4.2.3 Durchführung der Analysen

Modellsysteme berechnen im Modul 2 des AnTeS während Simulationen zu jedem Zeitpunkt selbstständig die Fehlerrate für die bisher insgesamt aufgetretenen Ausfälle des Gesamtsystems (also die bisher beobachtete Häufigkeit von „Failures on Demand“). Die Idee dahinter wird nachfolgend zunächst erläutert, bevor dann die Ansteuerung der Simulationen näher betrachtet wird.

Grundlegend wurde für die weiter unten vorgestellten Sensitivitätsanalysen angenommen, dass während der Simulation jederzeit ein Anforderungsfall vorliegt.

Dementsprechend müsste auch zu jedem Zeitpunkt während der Simulation am Ausgang des jeweils simulierten Systems dauerhaft eine logische 1 (für Auslösung) anliegen. Das Verhältnis der simulierten Zeit, in der aufgrund von aufgetretenen Fehlern kein Auslösesignal am Ausgang des Gesamtsystems ausgegeben wird, zur gesamten Simulationszeit liefert dann zu jedem Berechnungszeitpunkt eine Abschätzung der bisher aufgetretenen Wahrscheinlichkeit für das Versagen bei Anforderung („Failure on Demand“).

Das Post-Processing für die entsprechenden Berechnungen ist repräsentativ für das Modellsystem A222 in Abb. 4.13 dargestellt. Hierzu dient erneut ein Simulink-Subsystem (blauer Kasten in Abb. 4.13). Das Eingangssignal in das entsprechende Subsystem („In1“ bzw. „1“) wird zunächst in dessen Signalwert und das entsprechende Fehlerattribut zerlegt. Dies ist notwendig, da im Modul 2 des AnTeS in Anlehnung an die Signale des Moduls 1 des AnTeS (TXS) jedes Signal beide Informationen enthält. Für die Analysen im Rahmen dieses Vorhabens wurde dieses Fehlerattribut jedoch nicht weiterverwendet, da gemäß /PIL 18/ in allen Modellsystemen die jeweilige AL („Analog Logic“) keine entsprechenden Informationen ausgeben kann. Dementsprechend wird das Fehlerattribut in Abb. 4.13 auf einen sogenannten Terminator geleitet und damit defacto verworfen.

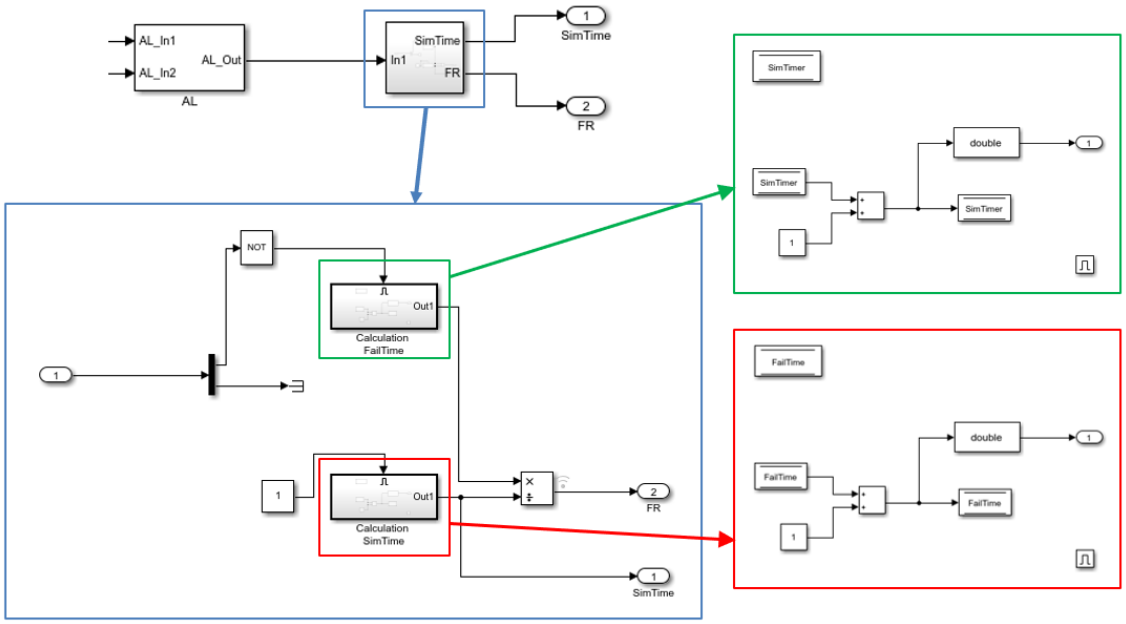


Abb. 4.13 Post-Processing im Modell A222 des AnTeS für die externe Ausgabe

Das in Abb. 4.13 im blauen Kasten gezeigte Subsystem enthält selbst zwei weitere Subsysteme.

Eines wird in jedem einzelnen Berechnungsschritt (mit definierter Zeitlänge) durch eine konstante 1 getriggert und hierüber die bis zum aktuellen Schritt insgesamt vergangene Zeit berechnet („Calculation SimTime“, roter Kasten), welche am Ausgang 1 („SimTime“) auch nach außen ausgegeben wird. Das zweite Subsystem wird nur dann getriggert, wenn gerade kein auslösendes Signal ansteht (da der Signalwert negiert wird – „NOT“). Auf diese Weise wird (da die Länge eines Zeitschritts per Definition bekannt ist), die bisher insgesamt vergangene Zeit berechnet, in der die Auslösung nicht stattfand („Calculation FailTime“, grüner Kasten). Die sich aus dem Verhältnis dieser Zeiten berechnete (beobachtete) Fehlerrate wird vom Modellsystem ebenfalls zu jedem Zeitschritt berechnet und über den Ausgang 2 (FR) ebenfalls ausgegeben.

Die Durchführung von Simulationsdurchläufen gestaltet sich dann wie folgt. Nach Erstellung des Modells in Simulink wird aus dem jeweiligen Modell eine DLL erstellt. Mit Hilfe eines eigenen Python-Skripts wird diese DLL geladen und diese die für den jeweiligen Durchlauf gültigen Parameter übergeben (siehe hierzu Anhang A.2). Im Beispiel des Modellsystems A222 sind das z. B. die 17 in diesem Simulationsdurchlauf gültigen Vergleichswerte („Wahrscheinlichkeiten“) von selbstmeldenden und nicht selbstmeldenden Fehlern, zwei Werte für die Abstände zwischen den WKPen in den Redundanzen 1 und 2 sowie die angenommene Dauer von Reparaturen.

Anschließend kann mit der Simulation (über für die DLL vorgegebene Steuerbefehle) jeweils ein Berechnungsschritt ausgeführt werden, wobei die von der DLL ausgegebenen Werte der bisherigen Fehlerrate („FR“) und der bisherigen Simulationszeit („SimTime“) erfasst werden. Zur Überwachung werden die aktuellen Ausgabewerte in vorgewählten Zeitabständen ausgegeben und gespeichert.

4.2.4 Ergebnisse

Im Modul 2 des AnTeS können aufgrund der Verwendung von DLLs (siehe auch Anhang A.2) Berechnungen sehr schnell durchgeführt und daher vergleichsweise lange simulierte Zeiträume betrachtet werden. Abb. 4.14 zeigt die aufgezeichneten Fehlerraten des Modellsystems A133 (vgl. hierzu auch Anhang A.3) als Funktion der Zeit für zehn Simulationsdurchläufe mit unterschiedlichen vorgegebenen Reparaturzeiten (im Sinne einer Sensitivitätsanalyse) für jeweils 10^6 Jahre. Nach jeweils etwa 10^5 Jahren stellen sich in diesem Beispiel bereits stabile berechnete Ausfallraten ein.

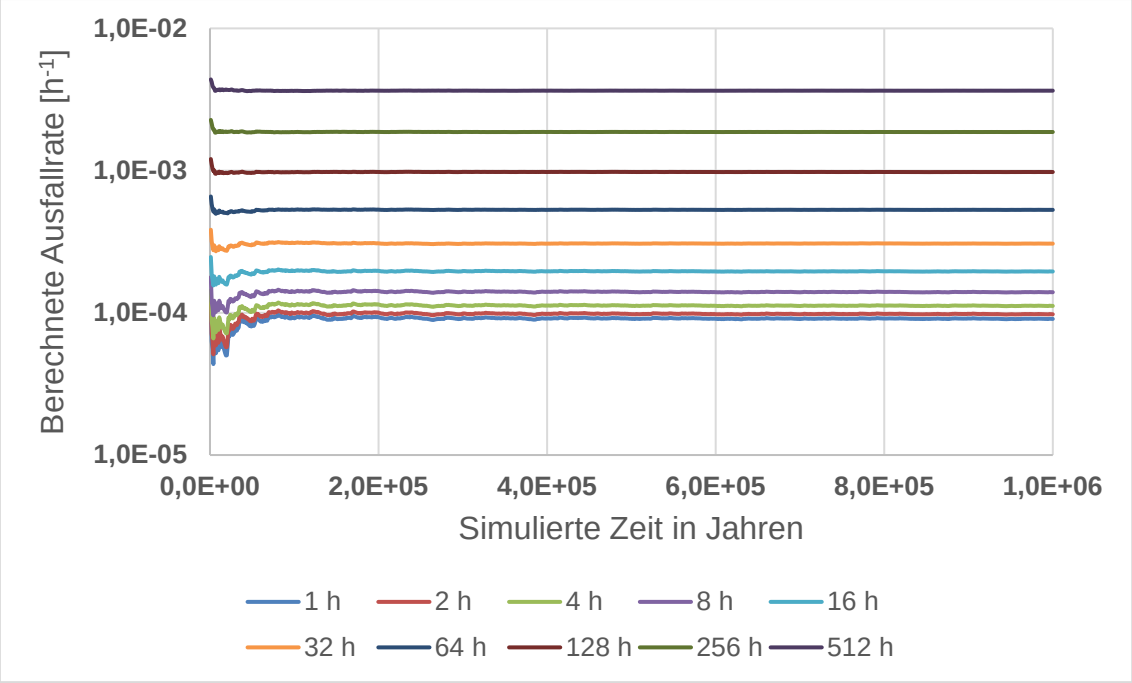


Abb. 4.14 Sensitivitätsanalyse hinsichtlich Reparaturzeiten für das Modellsystem A133 mit Hilfe des Moduls 2 des AnTeS

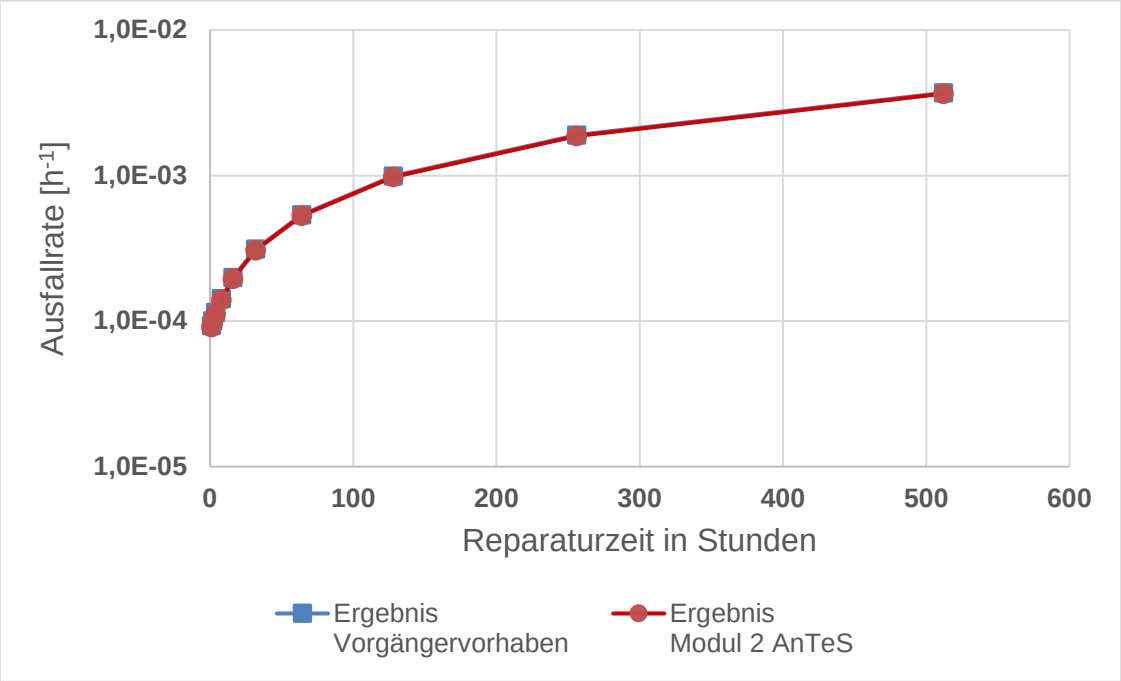


Abb. 4.15 Vergleich der Sensitivitätsanalysen für das Modellsystem A133 mit dem Modul 2 des AnTeS und im Vorgängervorhaben /PIL 18/.

In Abb. 4.15 sind die mit dem AnTeS bestimmten Ausfallraten für das Modellsystem A133 als Funktion der vorgegebenen Reparaturzeit dargestellt (Sensitivitätsanalyse).

Diese werden in der Abbildung zusätzlich mit den Ergebnissen aus dem Vorgängervorhaben verglichen. Innerhalb der erreichbaren Genauigkeit stimmen die Ergebnisse, welche mit dem Modul 2 des AnTeS durch Monte-Carlo-Simulationen bestimmt wurden, sehr genau mit den Ergebnissen aus dem Vorgängervorhaben (dort mit Hilfe von FMEAs und Fehlerbaumanalysen bestimmt) überein. Somit bestätigen einerseits die Berechnungen mit dem AnTeS die Ergebnisse des Vorgängervorhabens. Umgekehrt zeigt dies aber auch, dass die Berücksichtigung der Reparaturzeiten im Modul 2 des AnTeS funktioniert.

Dasselbe gilt auch für die Berücksichtigung der Abstände zwischen WKPen, wie in Abb. 4.16 für das Modellsystem A222 erkennbar. Auch die Ergebnisse dieser Sensitivitätsanalysen mit Hilfe des AnTeS und aus dem Vorgängervorhaben stimmen sehr gut überein.

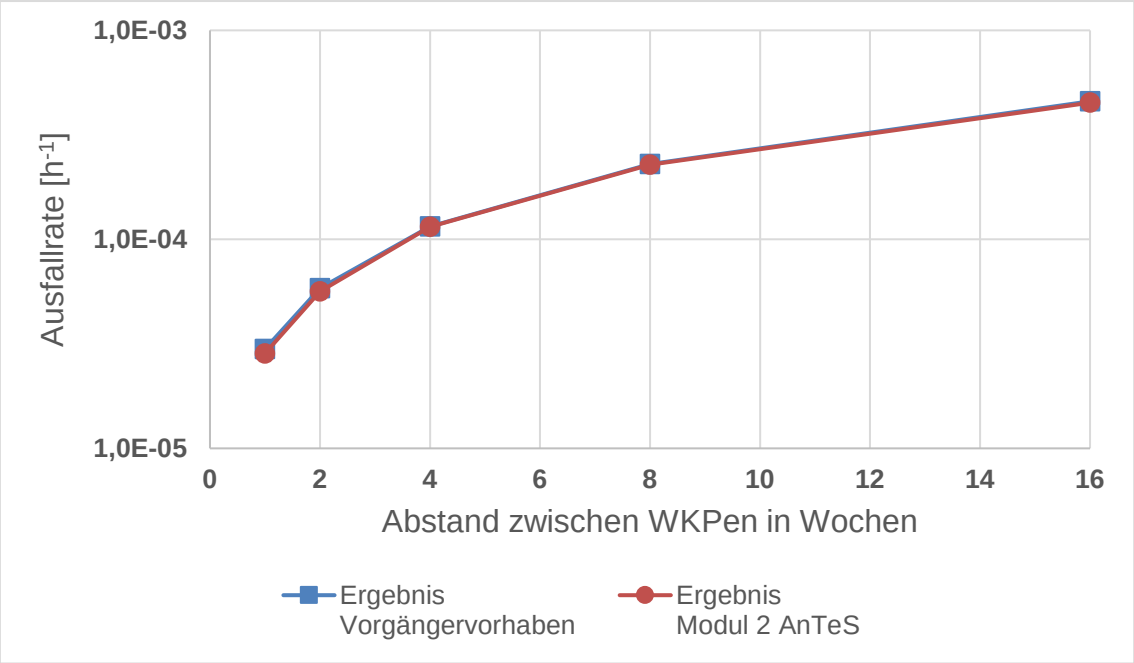


Abb. 4.16 Vergleich der Ergebnisse für das Modellsystem A222

Es ergeben sich allerdings geringfügige Abweichungen, wenn die bestimmten Gesamtausfallraten kleiner als in den bisherigen Beispielen werden. Dies wird anhand einer durchgeführten Sensitivitätsanalyse für das Modellsystem A222mod demonstriert. A222mod ist im Wesentlichen identisch mit dem Modellsystem A222. Es unterscheidet sich von diesem ausschließlich durch die Verwendung von „MAX“-Bausteinen anstatt von „2.MAX“-Bausteinen in den Funktionsplänen der PUs (siehe Abb. 4.17).

Hierdurch erhöht sich die Zuverlässigkeit des Modellsystems um etwa eine Größenordnung, d. h. die bestimmten Ausfallraten sind etwa eine Größenordnung kleiner (Details und insbesondere eine Erklärung hierzu können in /PIL 18/ nachgelesen werden).

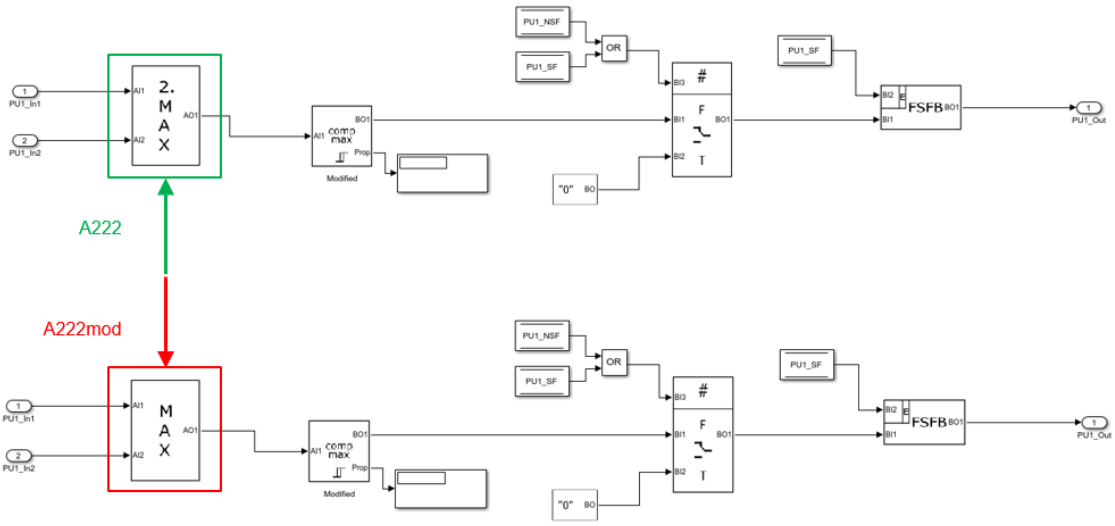


Abb. 4.17 Unterschied zwischen den Modellsystemen A222 und A222mod in den Funktionsplänen der PUs.

Die Ergebnisse der Sensitivitätsanalysen für das Modellsystem A222mod hinsichtlich der Abstände zwischen WKPen mit dem Modul 2 des AnTeS werden mit denen aus dem Vorgängervorhaben in Abb. 4.18 verglichen. Es zeigt sich ein Trend, dass die Ergebnisse des AnTeS für kleiner werdende berechnete Ausfallraten eine zunehmende Abweichung (nach unten) im Vergleich zu den Ergebnissen im Vorgängervorhaben haben.

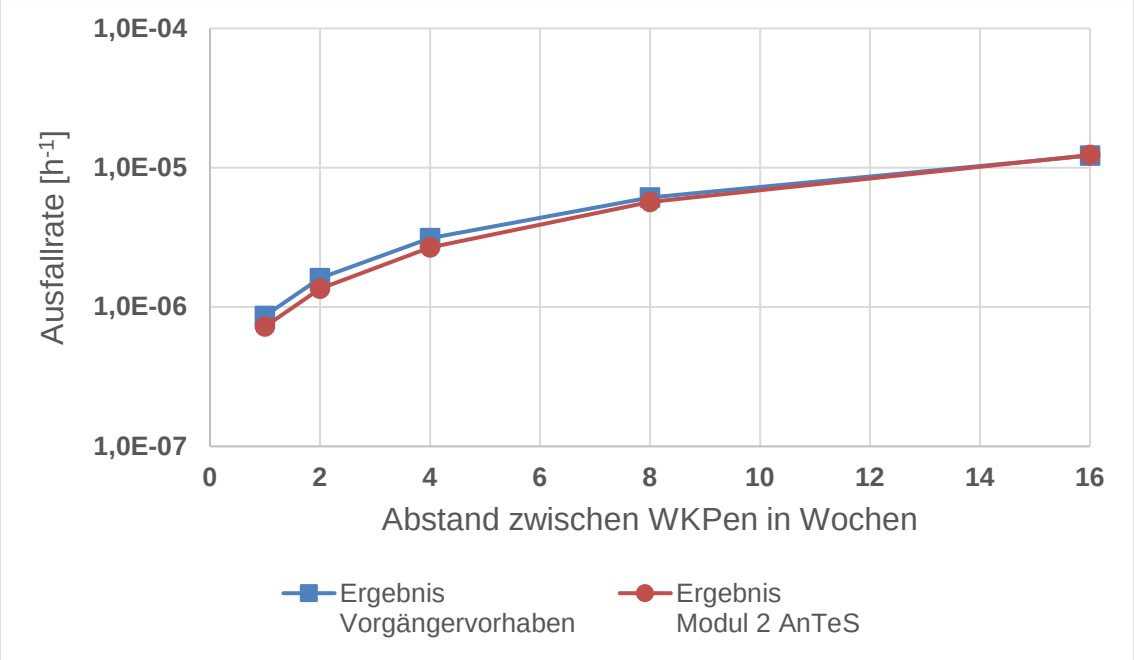


Abb. 4.18 Vergleich der Ergebnisse für das Modellsystem A222mod.

Eine Erklärung mit Bewertung dieser Abweichungen erfolgt im nachfolgenden Abschnitt, in welchem die Ergebnisse insgesamt noch einmal zusammengefasst und bewertet werden.

4.2.5 Bewertung der Ergebnisse

Die Sensitivitätsanalysen mit dem Modul 2 des AnTeS zeigen sehr ähnliche Ergebnisse für die Modellsysteme A222, A222mod und A133 im Vergleich zu den mit RiskSpectrum durchgeführten Analysen im Vorgängervorhaben. Für die Modellsysteme A133 und A222 sind diese sogar praktisch identisch (siehe Abb. 4.15 und Abb. 4.16). Für sehr kleine Ausfallraten ist aber eine sichtbare Abweichung der Ergebnisse (zu etwas kleineren Werten) vom Modul 2 des AnTeS im Vergleich zu den „klassischen“ Analysen erkennbar (für Werte < $1 \cdot 10^{-05}$ in Abb. 4.18). Diese konnten durch eine gezielte Analyse des Modells, im Sinne einer zusätzlichen Sensitivitätsanalyse, auf die begrenzte Auflösung der verwendeten digitalen Zahlenwerte innerhalb des für die Analysen verwendeten Modells zurückgeführt werden. Dieser Zusammenhang wird nachfolgend näher erläutert.

Fehlerraten werden im Modul 2 des AnTeS durch den Vergleich von Zufallszahlen mit einem vorgegebenen Vergleichswert berücksichtigt (vgl. Abb. 4.8).

Die Wertebereiche der Zufallszahlen umfassen dabei sehr große Intervalle (z. B. Zahlen zwischen 0 und 1 10^{10}). Diese werden für sehr kleine Fehlerraten mit vergleichsweise kleinen Zahlen verglichen. Geht man davon aus, dass für die Darstellung der Zufallszahlen (aufgrund der digitalen Darstellung) nur eine endliche Anzahl von Werten darstellbar ist, wird die „Auflösung“ für kleiner werdende Zahlen immer grober. Das soll schematisch durch Abb. 4.19 verdeutlicht werden. Nur zur Verdeutlichung wird in dieser Abbildung das Intervall 0 … $1 \cdot 10^5$ durch den Balken ganz links dargestellt. Aufgrund der Darstellung der Zahlen im Rechner als Fließkommazahl hängt die maximal erreichbare Auflösung im Intervall $[0,1] \times 10^x$ vom Exponenten x ab. Die minimalen Abstände der einzelnen Zahlen entsprechend der erreichbaren Auflösung werden in der Abbildung durch blaue Linien repräsentiert. Diese sind in im Balken ganz links noch so nah beieinander, dass diese den gesamten Bereich blau eingefärbt erscheinen lassen. Betrachtet man jetzt nur den Zahlenbereich zwischen 0 …$1 \cdot 10^4$ (zweiter Balken von links), so werden bereits die einzelnen Linien sichtbar. Für noch kleinere Teilintervalle (im Bild z. B. $1 \cdot 10^3$) liegen diese dann immer weiter „auseinander“.

Vergleicht man jetzt eine beliebe Zufallszahl aus dem Intervall 0, … $1 \cdot 10^3$ (blaue Linien im ganz rechten Balken in der Abbildung) mit einem (ebenfalls) sehr kleinen Vergleichswert („Abfrage im Modell: Zufallszahl < X“), so kommt es zu einer Abweichung nach unten, da maximal die nächstkleinere Zufallszahl berücksichtigt werden kann.

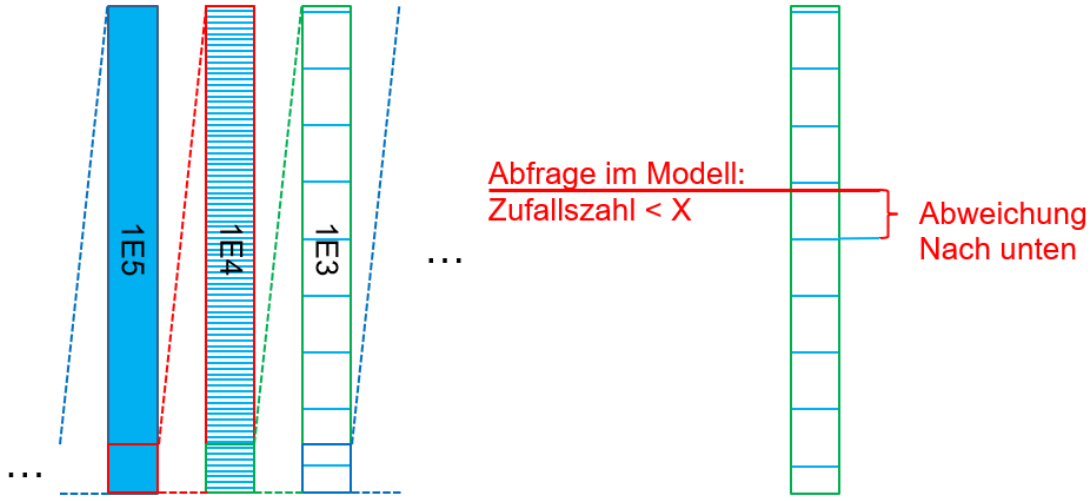


Abb. 4.19 Die begrenzte „Auflösung“ digitaler Zahlen kann zu systematischen Abweichungen führen.

Konkret wirkt sich das bei Simulationsergebnissen wie in Abb. 4.20 dargestellt aus. Hier wurden in einem eigens hierfür erstellten Modell (ähnlich zu Abb. 4.8) ausschließlich fünf Fehlerraten durch den Vergleich von Zufallszahlen mit Vergleichswerten gebildet und aufgezeichnet. Der Vergleichswert („Wahrscheinlichkeit“) wurde so gewählt, dass rechnerisch in allen Fällen eine Fehlerrate von $1 \cdot 10^{-9}$ h^{-1} erwartet werden kann. Zum einen ist in dem Diagramm zu erkennen, dass die zu einem bestimmten Zeitpunkt gerade berechnete Fehlerrate relativ stark schwankt (da nur noch sehr selten „Ereignisse“ auftreten). Zum anderen tritt auch eine systematische Abweichung nach unten auf. So liegt der Gesamtdurchschnitt aller beobachteten Fehlerraten (in diesem Beispiel) für $t > 10^5$ Jahren (rechte Hälfte) insgesamt bei $9{,}29 \cdot 10^{-10}$ h^{-1}, ist also um ca. 7 % kleiner als der angestrebte Wert ($1 \cdot 10^{-9}$ h^{-1}). Noch kleinere Fehlerraten lassen sich kaum noch berücksichtigen, da in diesem Fall nicht nur die beschriebene Abweichung relevant ist, sondern auch die Berücksichtigung noch seltenerer Ereignisse die notwendigen simulierten Zeiträume zu groß werden lassen, um diese auf aktuell üblichen PCs in akzeptabler Zeit zu berechnen.

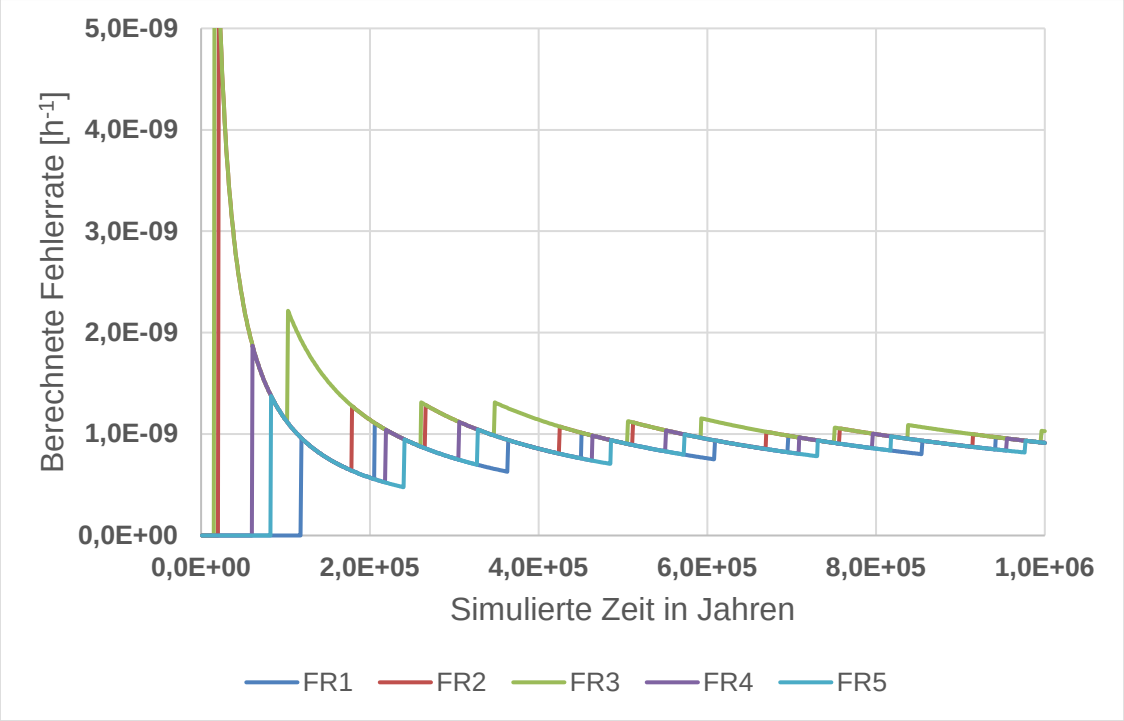


Abb. 4.20 Fehlerraten (Vergleichswert sollte $1 \cdot 10^{-9}$ h^{-1} ergeben) in fünf unterschiedlichen Simulationsdurchläufen (FR1, … FR5).

Gleichartige Versuche wurden für eine Reihe von eingestellten Fehlerraten durchgeführt, die Ergebnisse dieser Versuche sind in Tab. 4.2 zusammengefasst. Fehlerraten oberhalb von $1 \cdot 10^{-7}$ h^{-1} können demnach mit Hilfe von Simulink sehr genau berücksichtigt werden, relative Abweichungen vom gewünschten Wert sind dabei noch nicht erkennbar. Erst für noch kleinere Fehlerraten kommt es zu einer zunehmenden Abweichung, die aber auch erst unter ca. $5 \cdot 10^{-9}$ h^{-1} signifikant wird.

Tab. 4.2 Vergleich vorgegebener, mit beobachteten Fehlerraten (FR

Vergleich von vorgegebenen Fehlerraten (FR) mit beobachteten Fehlerraten (FR1, …, FR5) im Modul 2 des AnTeS (für ausreichend lange simulierte Zeiträume). In der letzten Spalte ist die relative Abweichung des Durchschnitts der beobachteten Fehlerraten vom angestrebten Wert dargestellt.

FR in h^{-1}	FR1 in h^{-1}	FR2 in h^{-1}	FR3 in h^{-1}	FR4 in h^{-1}	FR5 in h^{-1}	Mittelw. in h^{-1}	Rel. Abw.
1,000E-04	1,000E-04	1,000E-04	1,000E-04	1,000E-04	1,000E-04	1,000E-04	1,000
5,000E-05	5,000E-05	5,000E-05	4,999E-05	5,000E-05	5,000E-05	5,000E-05	1,000
1,000E-05	1,000E-05	1,000E-05	1,000E-05	1,000E-05	1,000E-05	1,000E-05	1,000
5,000E-06	5,002E-06	5,000E-06	5,002E-06	5,001E-06	4,999E-06	5,001E-06	1,000
1,000E-06	1,000E-06	9,990E-07	1,001E-06	1,000E-06	9,990E-07	9,998E-07	1,000
5,000E-07	5,000E-07	4,996E-07	4,997E-07	5,004E-07	4,998E-07	4,999E-07	1,000
1,000E-07	9,960E-08	1,000E-07	9,980E-08	9,980E-08	9,950E-08	9,974E-08	0,997
5,000E-08	4,983E-08	4,993E-08	4,967E-08	4,987E-08	4,980E-08	4,982E-08	0,996
1,000E-08	9,840E-09	9,840E-09	9,780E-09	9,890E-09	9,700E-09	9,810E-09	0,981
5,000E-09	4,651E-09	4,644E-09	4,655E-09	4,674E-09	4,615E-09	4,648E-09	0,930
1,000E-09	8,860E-10	9,370E-10	9,370E-10	9,710E-10	9,370E-10	9,336E-10	0,934
5,000E-10	4,684E-10	4,684E-10	4,684E-10	4,684E-10	4,684E-10	4,684E-10	0,937

Grundlegend muss derzeit daher empfohlen werden, nur Simulationen mit auftretenden Fehlerraten > $5 \cdot 10^{-9}$ h^{-1} durchzuführen. Zusätzlich sollte man bei Ergebnissen (für das Gesamtsystem) von unter ca. $1 \cdot 10^{6}$ h^{-1} davon ausgehen, dass diese um einige Prozent nach unten abweichen. Die Größe dieser Abweichung ist nämlich als umso größer anzunehmen, je mehr sehr kleine Fehlerraten einen signifikanten Einfluss auf das Gesamtergebnis haben.

Dieser Effekt lässt sich, z. B. durch die Wahl größerer Werteintervalle, nicht beheben, da letztlich immer die gleiche „Gesamtauflösung" der Zufallszahlen zum Tragen kommt. Das lässt sich verstehen, wenn man davon ausgeht, dass die Zufallszahlen in Simulink eigentlich immer im Intervall 0, …, 1 liegen und der gewählte Wertebereich quasi durch eine Multiplikation mit dem Maximalwert erfolgt. Zwar wird der Vergleichswert dann auch immer größer, aber eben auch die „Auflösung" schon für immer größere Werte schlechter.

Dieses Problem könnte folgender Ansatz vollständig beseitigen: Anstatt dem Modell von außen lediglich die *Vergleichswerte* (also die Wahrscheinlichkeiten) mitzuteilen, könnten die *Erwartungswerte* für das Auftreten der einzelnen Fehler extern berechnet werden und anschließend (zufällig über das simulierte Zeitintervall) über Fehlerinjektionen in das

System eingespeist werden. Dieses Vorgehen würde sicherstellen, dass die zu erwartenden Fehlerhäufigkeiten exakt gemäß ihrer Wahrscheinlichkeit auftreten würden. Für zuverlässige statistische Ergebnisse müssten dann nur noch geeignete simulierte Zeiträume gewählt werden, das Modell selbst würde dabei aber keine zu groben Zufallszahlen mehr verwenden. Dies könnte in einem späteren Vorhaben realisiert werden.

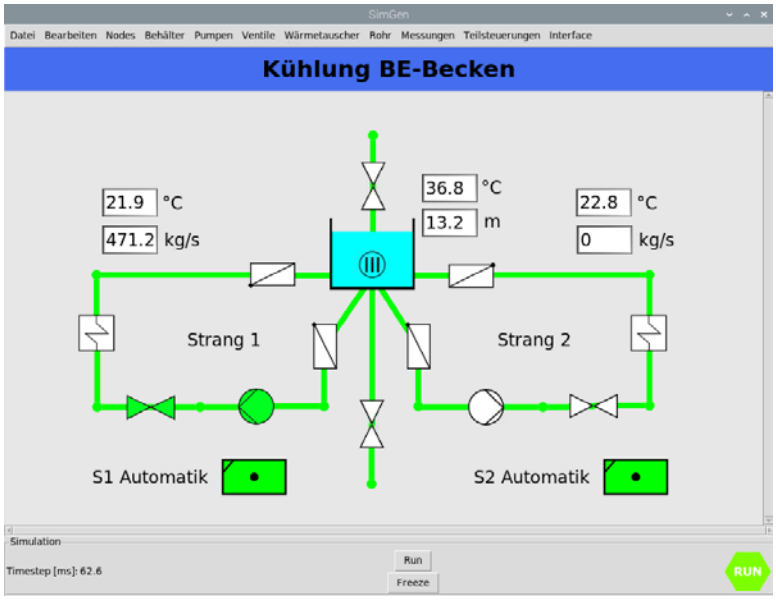
Insgesamt haben sich die Monte-Carlo-Simulationen, welche mit dem Modul 2 des AnTeS durchgeführt werden können, jedoch als ein äußerst wirksames neues Werkzeug erwiesen. Mit deren Hilfe konnten einerseits die Ergebnisse des Vorgängervorhabens sehr gut bestätigt werden, auf der anderen Seite zeigen diese Vergleiche auch, dass das AnTeS als zusätzliches Werkzeug für die Analyse von digitalen Leittechniksystemen nutzbringend eingesetzt werden kann.

4.3 Tests mit verfahrenstechnischen Simulationen

Wesentlich für die Weiterentwicklung und die Validierung der Methoden und Modelle der GRS zur Analyse von digitalen Leittechniksystemen waren die in den vorangegangenen Abschnitten vorgestellten Analysen mit dem Modulen 1 und 2 des AnTeS. Das Modul 3 des AnTeS erlaubt darüber hinaus insbesondere aber auch das Anbinden unterschiedlicher (verfahrenstechnischer) Simulationen an die Leittechniksysteme. Hieraus ergeben sich weitere interessante Möglichkeiten für Untersuchungen und Analysen, die in diesem Abschnitt anhand der Beschreibung durchgeführter Tests und Analysen näher beleuchtet werden.

4.3.1 Tests mit SimGen

Wie im Abschnitt 2.3.2 erläutert, können mit SimGen flexibel unterschiedlichste verfahrenstechnische (wasserführende) Systeme simuliert werden. Konkret wurden im Rahmen dieses Vorhabens im Wesentlichen drei unterschiedliche Simulationen mit SimGen erstellt und getestet: generische Simulationen einer Brennelement(BE)-Beckenkühlung und eines Forschungsreaktors (Poolreaktor) (vgl. Abb. 4.21) sowie eine simulierte Version von TeSys (vgl. Abb. 4.22), des aus realen Komponenten bestehenden (Prototyp des) Testsystems der GRS (vgl. Abschnitt 2.3.4).



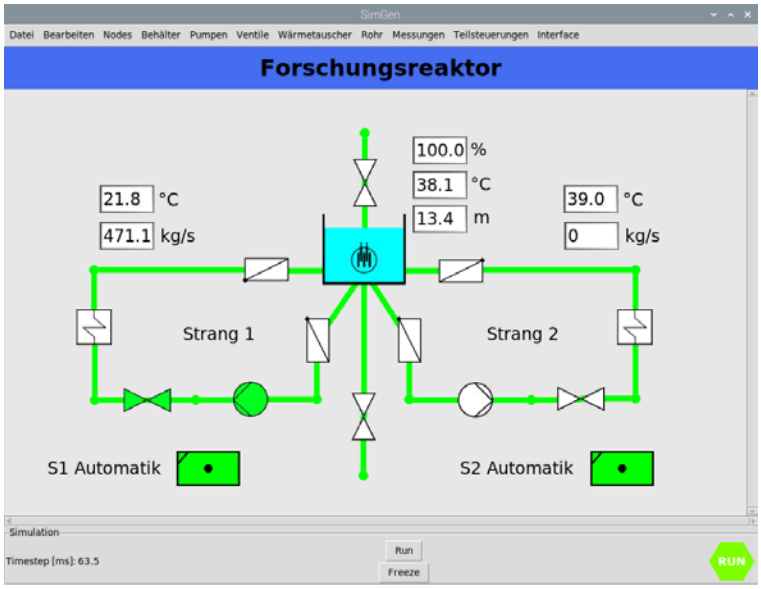


Abb. 4.21 Mit SimGen erstellte generische Simulationen einer BE-Beckenkühlung und eines Forschungsreaktors.

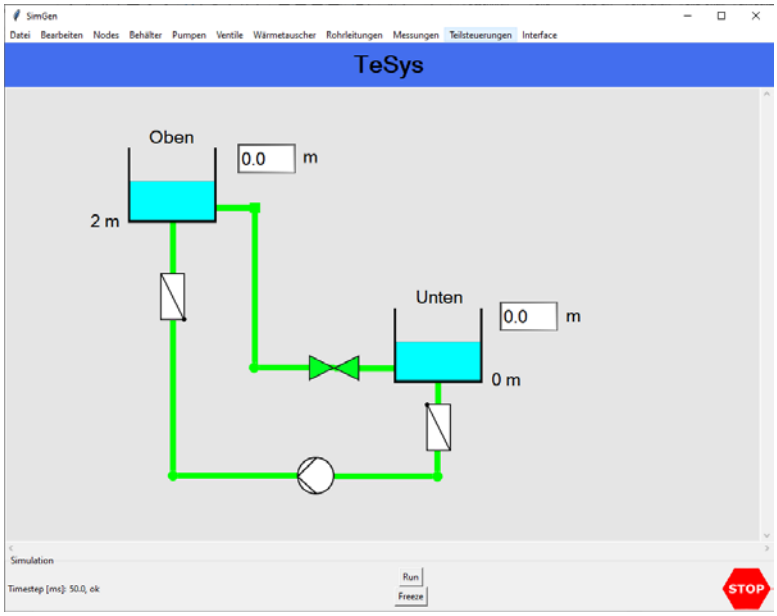


Abb. 4.22 Mit SimGen erstellte Simulation von TeSys (Testsystem der GRS, siehe Abschnitt 2.3.4).

Während mit der Simulation von TeSys nur einfache vergleichende Tests bezüglich des zeitlichen Verhaltens des verfahrenstechnischen Systems zur Erprobung von SimGen selbst durchgeführt wurden (durch Vergleich mit dem realen System), sind die generischen Simulationen der BE-Beckenkühlung und des Forschungsreaktors auch mit den Leittechniksystemen des AnTeS für integrierte Tests kombiniert worden. Nachfolgend wird dies repräsentativ am Beispiel der BE-Beckenkühlung etwas detaillierter erläutert.

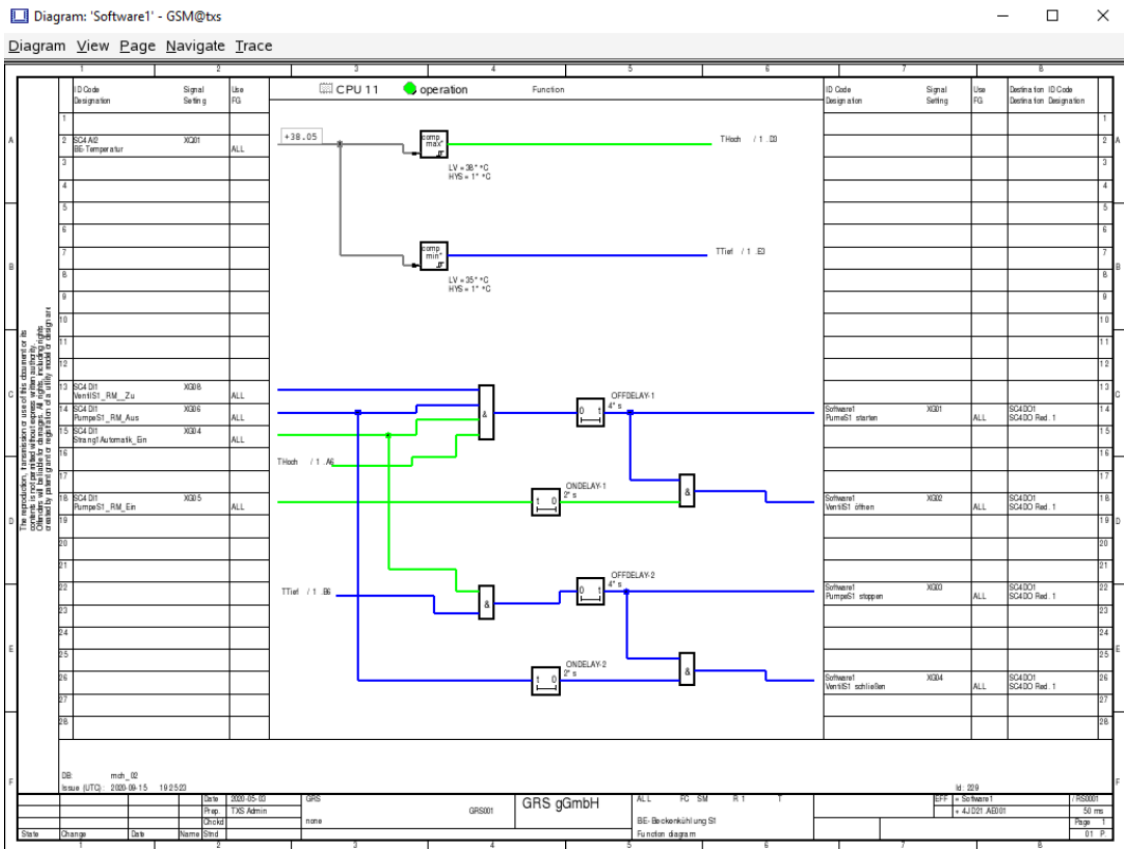


Abb. 4.23 Dynamischer Funktionsplan für den Strang 1 der BE-Beckenkühlung im graphischen Servicemonitor (GSM) auf dem Servicerechner des TXS

Abb. 4.23 zeigt einen dynamischen Funktionsplan für die automatische Steuerung des ersten Strangs der BE-Beckenkühlung im graphischen Servicemonitor (GSM) auf dem Servicerechner des TXS (Modul 1 des AnTeS). Ein analoger Funktionsplan, allerdings mit im Vergleich zum gezeigten Funktionsplan gestaffelten Grenzwerten, ist für die Steuerung des zweiten Strangs zuständig. Im oberen Drittel werden aus der gemessenen Temperatur im BE-Becken zwei unterschiedliche Grenzwerte für die Beckentemperatur gebildet (T_{Hoch} und T_{Tief}).

In den unteren zwei Dritteln des Funktionsplans werden aus den Grenzwerten Auslösesignale abgeleitet. So sorgt das obere UND (mit vier Eingängen) dafür, dass beim Überschreiten von T_{Hoch} zunächst die Pumpe des ersten Strangs gestartet wird (falls gleichzeitig die entsprechende Teilsteuerung eingeschaltet ist („Strang1 Automatik_Ein“), das Ventil im ersten Strang geschlossen („VentilS1_RM_Zu“ → Start nur gegen geschlossenes Ventil) und die Pumpe noch nicht läuft („PumpeS1_RM_Aus“). Das entsprechende Auslösesignal steht im Anforderungsfall für mindestens 4 s sicher an (OFFDELAY-1).

Meldet die Pumpe für mindestens 2 s, dass sie eingeschaltet ist (ONDELAY-1), so wird über das entsprechende UND nach dem Start der Pumpe das Ventil hinter der Pumpe geöffnet („VentilS1 öffnen“). Im unteren Drittel werden (bei Bedarf) analog die entsprechenden Signale zum Ausschalten der Pumpe und Schließen des Ventils gebildet, wenn die Temperatur im BE-Becken wieder ausreichend gesunken ist (unter T_{Tief}).

In der in Abb. 4.23 gezeigten Situation ist gerade, nachdem die Temperatur über T_{Hoch} angestiegen war, die Pumpe des ersten Strangs gestartet und das Ventil des ersten Strangs geöffnet worden. Die damit eingeschaltete Kühlung hat die Temperatur aber noch nicht wieder unter die Grenze T_{Hoch} (unter Berücksichtigung der gewählten Hysterese) gesenkt.

Mit den erläuterten Logiken wurden unterschiedlichste Testszenarien untersucht. So wurden beispielsweise verschiedene Fehler per Fehlerinjektion (vergleiche Abschnitt 3.1) ins System eingespeist und das beobachtete mit dem erwarteten Verhalten verglichen. Wird z. B. das Starten der Pumpe im ersten Strang verhindert, dann startet bei etwas höherer Temperatur automatisch der Kühlstrang 2 (aufgrund gestaffelter Grenzwerte) erwartungsgemäß und ohne signifikante zusätzliche Verzögerungen (aufgrund der Kopplung über das Interface). Alle weiteren Tests zeigten ebenfalls das jeweils erwartete Verhalten. Insbesondere wurde dadurch auch nachgewiesen, dass mit SimGen erstellte verfahrenstechnische Simulationen, wenn sie mit den Leittechniksystemen des AnTeS gekoppelt werden, das Verhalten des Gesamtsystems adäquat abbilden und damit unterschiedlichste Analysen durchgeführt werden können.

4.3.2 Tests mit KranSim

Prinzipiell gelten bei der Verwendung von KranSim im Zusammenspiel mit dem AnTeS dieselben Aussagen, wie bei der Verwendung des AnTeS mit SimGen. Nur, dass in diesem Fall das angesteuerte System keine Simulation eines wasserführenden verfahrenstechnischen Systems ist, sondern die Simulation eines Maschinenhauskrans. In Kombination mit den Leittechniksystemen des AnTeS lässt sich daher ein Gesamtsystem aus einem simulierten Maschinenhauskran mit automatischen Steuerungs- und Sicherheitsfunktionen umsetzen und untersuchen. Darüber hinaus wurden in KranSim aber auch ein paar spezielle automatische Analysemöglichkeiten direkt in die Software integriert. Diese besonderen Analysemöglichkeiten waren das Ziel der in diesem Abschnitt beschriebenen Tests.

Abb. 4.24 zeigt das Fenster von KranSim, in welchen Einstellungen für automatische Analysen vorgenommen werden können und auf welchem diese Analysen auch unmittelbar gestartet werden können. Prinzipiell stehen drei unterschiedliche Analysemöglichkeiten zur Verfügung:

- Allgemeine Analyse des Bewegungsraums,
- Lastabsturzanalyse im gesamten Bewegungsraum und
- spezielle Analysen entlang eines definierten Bewegungspfads

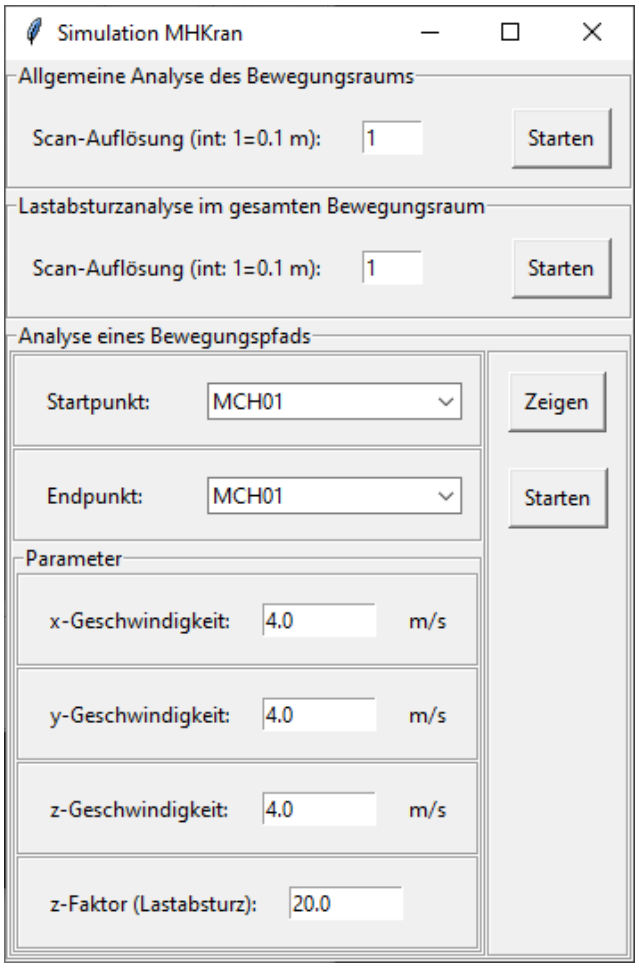


Abb. 4.24 Steuerung und Einstellungen von automatischen Analysen in KranSim

Für eine einfache, allgemeine Analyse des Bewegungsraums (also dem Innenraum des Maschinenhauses inklusive aller Hindernisse), muss ganz oben im Fenster im entsprechenden Eingabefeld lediglich eine Scan-Auflösung eingestellt werden und über den Button „Starten“ die entsprechende Analyse gestartet werden.

Bei der allgemeinen Analyse des Bewegungsraums rastert die Software KranSim den ganzen Bewegungsraum mit der aktuell eingestellten Last (derzeit können hier nur Würfel und Kugeln unterschiedlicher Größe als Last verwendet werden) ab.

Die Last nimmt nacheinander automatisch jede Position im gesamten Innenraum des Maschinenhauses ein, wobei automatisch getestet und aufgezeichnet wird, ob in der jeweiligen Position eine Kollision (formal ein Überlappen der Last mit einem Hindernis) vorliegt. Nach der vollständigen Rasterung des Bewegungsraums, die während der Analyse im 3D-Fenster beobachtet werden kann, liegt dann eine Zahl vor, die sich aus dem Verhältnis von Positionen mit Kollision zu Positionen insgesamt berechnet. Diese Zahl kann als ein einfacher Indikator dafür dienen, wie „gefährlich“ das Bewegen einer Last in diesem Bewegungsraum ist. Entfernt man beispielsweise alle Hindernisse, so ergibt sich eine 0 (keine Kollision im gesamten Bewegungsraum), wäre der gesamte Bewegungsraum dahingegen mit Hindernissen vollständig gefüllt, so ergibt sich eine 1 (Kollision in jeder Position). Realistische Verteilungen von Hindernissen liefern dementsprechend Zwischenwerte (zwischen 0 und 1), die in etwa die Füllung des Bewegungsraums mit Hindernissen widerspiegeln. Die Ausgabe von Ergebnissen erfolgt in der aktuellen Version von KranSim noch auf der Konsole (siehe Abb. 4.25).

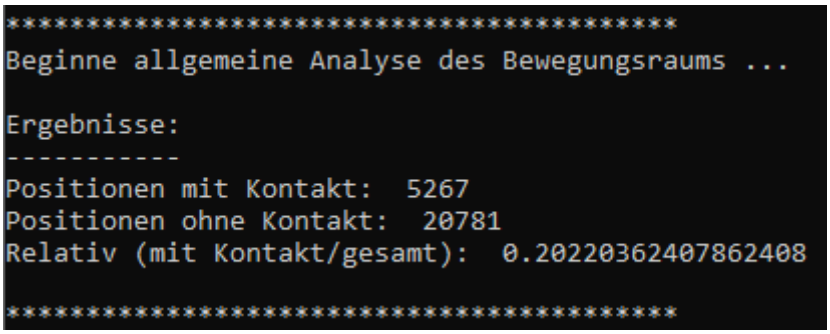


Abb. 4.25 Beispielergebnis einer allgemeinen Analyse des Bewegungsraums mit KranSim

In diesem Beispiel wurden etwa in 20% des gerasterten Innenraums Kollisionen detektiert.

Einen ähnlichen Wert würde man durch den Vergleich des mit Hindernissen gefüllten Innenraums und seinem gesamten Volumen erhalten, wobei dann jedoch besondere Kollisionen (z. B. des Seils beim Unterfahren von Auslegern o. ä.) nicht berücksichtigt würden. Diese allgemeine Analyse liefert also einen einfachen Indikator, der einen grundsätzlichen Vergleich von unterschiedlichen Konfigurationen des Bewegungsraums erlaubt.

Genähert kann aus dem Vergleich zweier Zahlenwerte für unterschiedliche Verteilungen von Hindernissen im Bewegungsraum auf den quantitativen Unterschied der Wahrscheinlichkeiten für unerwünschte Kollision geschlossen werden.

Eine ähnliche Analyse kann über „Lastabsturzanalysen im gesamten Bewegungsraum“ gestartet werden. Im Unterschied zur allgemeinen Analyse des Bewegungsraums wird bei dieser Art der Analyse jedoch die Last in jede x- und y-Position (mit der gewählten Scan-Auflösung) auf maximaler Höhe (maximales z) gebracht und anschließend fallengelassen und die dabei aufgetretenen Kollisionen aufgezeichnet. Auch hier liefert die Analyse in Abhängigkeit der vorhandenen Hindernisse (quantitative, vergleichbare) Ergebnisse zur allgemeinen „Gefährlichkeit“ (in diesem Fall hinsichtlich Lastabstürzen). Ein Beispielergebnis dieser Analysemöglichkeit ist in Abb. 4.26 zu sehen.

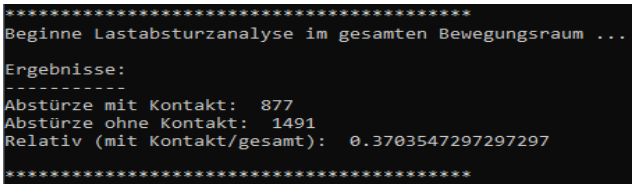


Abb. 4.26 Ergebnis einer Lastabsturzanalyse

In diesem Beispiel wurden in 37 % aller (x-, y-)Positionen bei einem Lastabsturz Kollisionen mit Hindernissen festgestellt.

Sowohl die allgemeine als auch die Lastabsturz-bezogene Analyse des Bewegungsraums kann je nach eingestellter Scan-Auflösung typischerweise innerhalb einiger Sekunden bis Minuten durchgeführt werden. Wobei bereits eine relativ grobe Scan-Auflösung (z. B. bei einem Wert von „10“ (~ 1-m-Raster)) zuverlässige Ergebnisse liefert, d. h. wesentlich feinere Scans liefern keine signifikant besseren Ergebnisse mehr.

Derzeit sind nur Analysen von Bewegungspfaden zwischen zwei Punkten möglich. Die Analyse von komplizierteren Pfaden erfordern daher derzeit noch mehrere Analysedurchläufe für die Einzelpfade. Für Analysen von Bewegungspfaden zwischen zwei Punkten müssen ein Start- und ein Endpunkt ausgewählt werden. Entsprechende Positionen können einfach händisch angefahren und unter einem frei wählbaren Namen gespeichert werden (über den Button „Positionen“ auf dem Steuerungsfenster von KranSim). Nach der Auswahl der Positionen kann über „Zeigen“ der so ausgewählte Pfad betrachtet werden (der Kran fährt diesen dann einmal ungestört ab). Über die entsprechenden Eingabefelder können für den zu analysierenden Pfad noch die Krangeschwindigkeiten in x-, y- und z-Richtung (jeweils in m/s) eingestellt werden.

Der zusätzliche Einstellwert „z-Faktor Lastabsturz“ beschleunigt entsprechend des gewählten Werts bei der eigentlichen Analyse den Ablauf des Lastabsturzes, um die für die Analyse benötigte Zeit zu verkürzen. Der vorgewählte Wert von 20 für diesen Parameter hat sich bewährt. Zu große Beschleunigungsfaktoren können aufgrund der dann auftretenden „Sprünge“ zwischen zwei Berechnungsschritten zu unerwünschten Ergebnissen führen (z. B. zu Sprüngen „durch“ den Boden).

Bei der eigentlichen Analyse des Bewegungspfades wird dieser schrittweise abgefahren und in jeder Position getestet, wie sich ein Lastabsturz an dieser Position auswirken würde. Zusätzlich wird anschließend der zu untersuchende Pfad in die Bereiche aufgeteilt, zwischen denen Steuerungsbefehle notwendig sind und die Auswirkungen beim Ausbleiben von diesen Steuerbefehlen untersucht. Zu diesem Zweck wird in den entsprechenden Positionen die Bewegung mit allen dann noch bestehenden Bewegungen (in x-, y- und z-Richtung) fortgesetzt bis entweder der Boden, ein Hindernis oder eine „Außenwand“ des Bewegungsraums getroffen wird (bzw. der Bewegungsraum verlassen wird).

Verdeutlicht wird dies durch das Beispielergebnis in Abb. 4.27. Hier wurde ein Pfad getestet, für den zwischen den Schritten 0 (Start) und 36 („0+“), 36 und 86 („36+“), 86 und 344 (Ende) („86+“) Richtungsänderungen notwendig sind (diese ergeben sich aus den eingestellten Geschwindigkeiten für die einzelnen Richtungen). In diesem Beispiel würden in ~76 % der Positionen bei Lastabstürzen der Boden, in ~24 % der Positionen Hindernisse getroffen werden. Durch ein Steuerungsversagen würden in ca. 90 % der Positionen Situationen entstehen, die eine Auslösung über einen der Wegendschalter („BR verlassen“; BR – Bewegungsraum) notwendig machen würden, während es in den weiteren 10 % der Positionen zu einer Kollision kommen würde (die dann auch durch Schutzfunktionen nicht verhindert würden). Aus diesen Zahlen kann (näherungsweise) auf Wahrscheinlichkeiten geschlossen werden, falls z. B. aus anderer Quelle die Wahrscheinlichkeit für einen Steuerungsversagen bekannt ist.

Nimmt man z. B. an, dass bei Kranbewegungen eine Fehlerrate von 10^{-4} h^{-1} für ein Steuerungsversagen aus anderer Quelle bekannt ist, so ergibt sich näherungsweise die folgende (nicht repräsentative) Wahrscheinlichkeit $p_{Kollision}$ für eine Kollision mit einem Hindernis (auf diesem Pfad):

$$p_{Kollision} = 10^{-4}\ h^{-1} \cdot t_{Pfad} \cdot A_{Hindernis}$$

mit

t_{Pfad} – Zeit für das Abfahren des Pfades

$A_{Hindernis}$ – Anteil des Pfades, auf dem Steuerungsversagen zu einer Kollision mit einem Hindernis führt

In Zahlen:

$$p_{Kollision} \approx\ 10^{-4}\ {}^{1}/_{h} \cdot \frac{344 \cdot 0{,}08\ m}{4 \cdot 3600\ {}^{m}/_{h}} \cdot 0{,}105 \approx 2{,}0 \cdot 10^{-8}$$

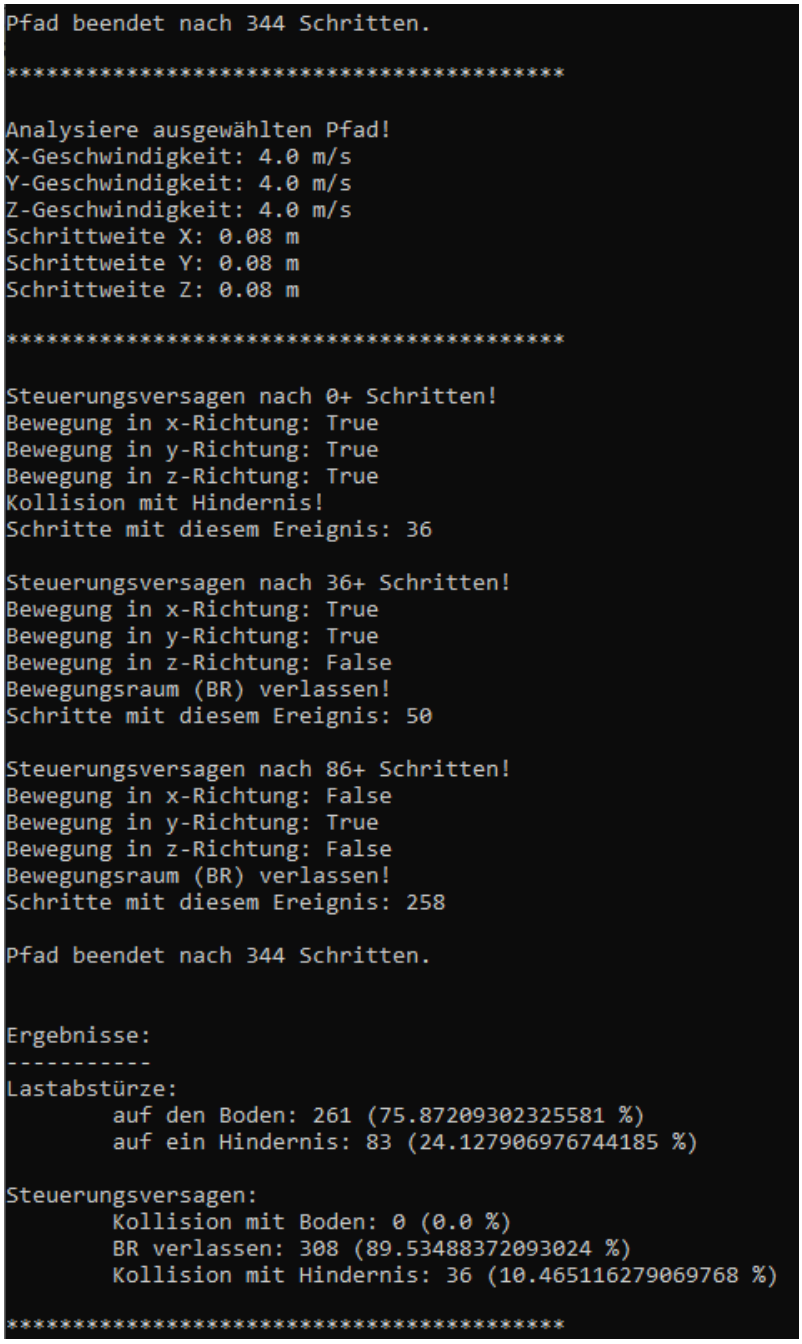


Abb. 4.27 Beispielergebnis für eine Analyse entlang eines definierten Pfads in KranSim.

KranSim kann (mit Hilfe von Erkenntnissen z. B. aus der Betriebserfahrung) also auch quantitative Ergebnisse über reine Vergleichswerte hinaus liefern. Das Ziel der Entwicklung von KranSim war es, bei der Bewertung der Wirksamkeit einzelner Funktionen von Kransteuerungen potentielle Gefährdungen abzuschätzen. Die Tests mit KranSim zeigen, dass dies mit dem neu entwickelten Tool geleistet werden kann.

5 Zusammenfassung und Ausblick

Im Rahmen dieses Vorhabens (4718R01314 – „Forschungsarbeiten zur Weiterentwicklung der Methode der Sensitivitätsanalyse zur Bewertung von Fehlerauswirkungen auf ein Leittechnik-Testsystem“) wurden neue Forschungswerkzeuge und Methoden zur Untersuchung der Fehlerauswirkungen in sicherheitsrelevanter digitaler Leittechnik entwickelt. Ein Grundstein dieser Entwicklungen war die Konzeption und der Aufbau des Analyse- und Testsystems (AnTeS) der GRS. Für dessen Aufbau wurde ein reales Leittechniksystem (Hard- und Software eines Ausrüsters von Kernkraftwerken) am Standort Garching der GRS aufgebaut und in Betrieb genommen (Modul 1 des AnTeS). Zusätzlich wurde auch die Möglichkeit geschaffen, sowohl Funktionen dieses realen Leittechniksystems als auch Funktionen generischer Leittechniksysteme zu simulieren (Modul 2 des AnTeS). In beiden Fällen können nicht nur Eingangssignale frei eingestellt und Ausgangssignale ausgelesen werden, sondern auch, ebenfalls im Rahmen der hier vorgestellten Arbeiten entwickelte, verfahrenstechnische Simulationen angekoppelt werden (Modul 3 des AnTeS). Alternativ können im Modul 3 des AnTeS aber auch beliebige verfahrenstechnische Prozesse angebunden werden, wie mit einem eigens hierfür erstellten Prototyp des Testsystems TeSys (bestehend aus realen Komponenten: u. a. Pumpen, Antriebe, Ventile, Instrumentierung, Stromversorgung) demonstriert werden konnte.

Im Rahmen von Arbeiten am AnTeS wurde auch eine Reihe neuer Analysewerkzeuge entwickelt, die z. B. auch die Validierung und Verifizierung (V&V) digitaler Leittechnikfunktionen und -systeme ermöglichen. So können sowohl in das reale als auch in ein simuliertes Leittechniksystem Fehler für Testzwecke injiziert werden (Fault-Injection). Hierdurch sind sowohl automatisierte Fehlerauswirkungsanalysen als auch Monte-Carlo-Simulationen mit Hilfe der simulierten Leittechniksysteme durchführbar. Durch die Anwendung der neuen Werkzeuge konnten die Ergebnisse und Aussagen des Vorgängervorhabens validiert werden.

Zusätzliche Tests demonstrieren darüber hinaus, dass mit dem AnTeS und allen im Vorhaben entwickelten Werkzeugen und Methoden Forschungsarbeiten und Analysen möglich werden, die weit über die Ziele dieses Vorhabens hinaus sachdienlich eingesetzt werden können.

So wird das AnTeS beispielsweise bereits in einem weiteren GRS-Vorhaben (4719R01374) zur Entwicklung einer Methode zur Bewertung von Vorkehrungen gegen potentielle Fehler bei Softwareänderungen in sicherheitsrelevanter Leittechnik (MEDIC-Method for Evaluation of HMI of Digital Instrumentation & Control /GRS 21/) eingesetzt.

Durch die hohe Flexibilität des AnTeS eröffnen sich vielfältige Möglichkeiten für weitere Forschungsarbeiten. Einige Möglichkeiten werden exemplarisch nachfolgend stichpunktartig aufgelistet:

- Erweiterung der Analysebreite durch Einbindung eines betrieblichen Leittechniksystems (BELT):

    - Inbetriebnahme und Kopplung mit dem AnTeS von Simatic-S7-Hard- und -Software (bei der GRS vorhanden) und damit
    - Erweiterung des AnTeS hinsichtlich Berücksichtigung von Funktionen der Betriebsleittechnik (BELT) und von Wechselwirkungen zwischen Sicherheitsleittechnik (SILT) und BELT.

- Erweiterung der Analysemöglichkeiten durch Einbindung digitaler Einrichtungen der Vorrangebene moderner Leittechniksysteme:

    - U. a. der Prioritätsmodule AV42, SPLM1-PC11 (TXS) (bei der GRS vorhanden),
    - Untersuchungen möglicher Ausfälle der Vorrangfunktionen: Sicherstellung des Vorrangs der Signale aus der Sicherheitsleittechnik vor betrieblichen Befehlen, Berücksichtigung der Rückmeldungen von Komponenten, Befehlsabsteuerung bei Schwergängigkeit (Drehmomentschutz) und in den Endlagen, Ansteuerung eines Antriebes aus zwei Bereichen.

- Untersuchungen zu verschiedenen Automatisierungssystemen und -einrichtungen (z. B. SILT und BELT) von:

    - Kränen und Hebezeugen,
    - Forschungsreaktoren und
    - Systemen im Nachbetrieb (z. B. BE-Becken).
    - Das AnTeS erlaubt integrale Tests und Analysen mit dem jeweiligen Gesamtsystem (bestehend jeweils aus (simulierten) verfahrenstechnischen System mit angekoppeltem Steuerungssystem)

- Weiterentwicklung des verfahrenstechnischen Testsystems (TeSys):

- Aufbau eines flexiblen Testsystems aus realen Komponenten und dadurch Erstellung eines weiteren Forschungswerkzeugs.
  - Damit z. B. auch: Überprüfung des dynamischen Verhaltens von Ventilantrieben unter spezifizierten Bedingungen (u. a. Stresstest),
  - Untersuchung der aus der Betriebserfahrung bekannten Ausfälle an der Schnittstelle zwischen Leittechnik und Antriebssteuerung, u. a. Fehlfunktionen der Wege-, End- und DREHMO-Schalter.
- Ein entsprechender Prototyp wurde im Rahmen dieses Vorhabens bereits aufgebaut und getestet.
- TeSys stellt ebenfalls eine Ergänzung des AnTeS dar.

Die Arbeiten in diesem Vorhaben haben die Kompetenzen der GRS hinsichtlich der Bewertung der Zuverlässigkeit digitaler, d. h. programmierbarer und rechnerbasierter, Leittechniksysteme deutlich erweitert, was der Mitarbeit in Gremien und anderen Forschungsarbeiten im Fachgebiet Leittechnik dient.

6 Referenzen

/BLE 21/ Blender ist eine freie, mit der GPL lizenzierte Software https://www.blender.org/, zuletzt abgerufen: 01.03.2021.

/BMU 15/ Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit Änderungen und Neufassung der Bekanntmachung zu den „Sicherheitsanforderungen an Kernkraftwerke“
BAnz AT 30.03.2015 B2, März 2015

/BMU 15a/ Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit: Bekanntmachung der Interpretationen zu den Sicherheitsanforderungen an Kernkraftwerke. BAnz AT 30.03.2015 B3, März 2015

/DIG 21/ Projekt der NEA/CSNI/WGRISK, Digital I&C PSA – Comparative Application of Digital I&C Modelling Approaches for PSA, Abschlussbericht in Vorbereitung, März 2021.

/FRW 06/ Frey, W. et al.:Erprobung und Bewertung der Methoden einer PSA für SWR-Anlagen der Baulinie 69 nach Stand von Wissenschaft und Technik (PSA SWR 69). Gesellschaft für Anlagen- und Reaktorsicherheit, Fachbände GRS-A-3292, GRS-A-3293, Garching, 2006.

/GRS 21/ Laufendes Vorhaben, BMU-Förderkennzeichen 4719R01374: Entwicklung einer Methode zur Bewertung von Vorkehrungen gegen potentielle Fehler bei Softwareänderungen sicherheitsrelevanter Leittechnik

/HEJ 15/ Herb, J. et al.: Entwicklung eines Ansatzes zur Analyse der Netzwerktechnologien in sicherheitsrelevanten Leittechniksystemen hinsichtlich Verbreitung und Auswirkung postulierter Fehler. Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Bericht GRS-377,
ISBN 978-3-944161-58-7, Juni 2015.

/MAT 21/ MATLAB und das Zusatzprodukt SIMULINK sind kommerzielle Software von MathWorks (Sitz in Massachusetts, USA), https://de.mathworks.com/, zuletzt abgerufen: 01.03.2021.

/NEA 15/ NEA/CSNI: Failure Modes Taxonomy for Reliability Assessment of Digital Instrumentation and Control Systems for Probabilistic Risk Analysis Nuclear Safety. NEA/ CSNI/ R(2014)16, February 2015.

/PAN 21/ Panda3D ist eine freie Software unter BSD-Lizenz (Open Source), https://www.panda3d.org/, zuletzt abgerufen: 01.03.2021

/PIL 18/ Piljugin, E. et al.: Entwicklung und Erprobung eines Werkzeugs zur Sensitivitätsanalyse der Fehlerauswirkungen in der sicherheitsrelevanten digitalen Leittechnik. Bericht: GRS - 494, ISBN 978-3-946607-79-3, März 2018.

/RIS 21/ RiskSpectrum ist eine kommerzielle Software von Lloyd's Register (Sitz London, UK), https://www.lr.org/en/riskspectrum/technical-information/psa/, zuletzt abgerufen: 01.03.2021.

/TXS 12/ Areva (Anm.: heute Framatome): Teleperm XS User Manual, TXS Core-SW Release 3.6.5, PTLSD-G/2012/en/0217 A (Restricted), 2012

/WIK 21/ Wikipedia – Sortierverfahren, https://de.wikipedia.org/wiki/Sortierverfahren, zuletzt abgerufen: 01.03.2021.

7 Abbildungsverzeichnis

Abb. 2.1 Das Analyse- und Testsystem (AnTeS) der GRS 3

Abb. 2.2 Anlieferung des Teleperm-XS-Systems ab Mai 2017 bei der GRS in Garching (linkes Bild). 5

Abb. 2.3 Modul 1 des AnTeS 5

Abb. 2.6 Der Simulink Library Browser zur Auswahl von Funktionsblöcken 8

Abb. 2.8 Interne Logik (Simulink-Subsystem) des Funktionsblocks „2. MAX“ in Abb. 2.7. 12

Abb. 2.9 Detailansicht der internen Logik des Bausteins „2. MAX“. 13

Abb. 2.10 Beispiel für die Sortierung der vier Eingangswerte im Funktionsblock „2.MAX“ (Detailansicht aus Abb. 2.8). 14

Abb. 2.11 Zwei repräsentative Beispiele für den Vergleich von {1, 8} bzw. {8, 1} mit einem Subsystem innerhalb der Logik des „2. MAX“-Bausteins. 14

Abb. 2.12 Bestimmung der Anzahl der validen Eingangssignale im Funktionsbaustein „2. MAX“. 15

Abb. 2.14 Eines der beiden AnTeS-Interfaces in einem der beiden TXS-Schränke 18

Abb. 2.15 Front- und Rückseite eines AnTeS-Interfaces im geöffneten Zustand 20

Abb. 2.16 Schaltplan der AnTeS-Interfaces 21

Abb. 2.17 Software der Interfaces zum Generieren von Eingangssignalen und zum Auslesen von Ausgangssignalen des Leittechniksystems (Modul 1 des AnTeS). 22

Abb. 2.18 Die Software SimGen der GRS zur Erstellung verfahrenstechnischer Simulationen 23

Abb. 2.19 Zwei Detailbilder von der Software KranSim, die von der GRS zur Simulation eines Maschinenhauskrans erstellt wurde. 25

Abb. 2.20 Übersichtsschaltbild des TeSys-Prototyps 27

Abb. 2.21 Schaltschrank, Antrieb mit angeschlossenem Ventil und Antriebssteuerungsrack vom TeSys 29

Abb. 2.22 Funktionsplan zur Bildung von Grenzwerten für Tests mit dem TeSys. 30

Abb. 2.23 Funktionsplan für die Bildung von Auslösesignalen bei Tests mit TeSys 31

Abb. 2.24 TeSys in Betrieb 32

Abb. 3.1 Gezielte Fehlersteuerung eines Signals in einem Funktionsplan. Der ungestörte Signalverlauf ist grün gekennzeichnet. 38

Abb. 3.2 Funktionsplan zur Untersuchung des Verhaltens eines UND-Funktionsblocks. 38

Abb. 3.3 Beispiel für eine Fehlerinjektion in das Ausgangssignal eines Grenzwertbausteins („comp“) im Modul 2 (Matlab/Simulink) des AnTeS. 39

Abb. 3.4 Die Steuerung der ersten beiden von 128 möglichen Fehlersignalen..... 41

Abb. 3.5 Setzen eines Speicherbaustein (RS-Flipflop-Funktionsblock) des ausgewählten Fehlersignals 43

Abb. 3.6 Setzen eines Speicherbaustein (RS-Flipflop-Funktionsblock) des ausgewählten Fehlersignals (inaktiv) 44

Abb. 3.7 Eingabefenster des Programms „Fehlerkombinationsauswirkungs-Analyse“ 46

Abb. 3.8 Im Programm „Fehlerkombinationsauswirkungs-Analyse“ vorgenommene Einstellungen für die im Text beschriebene Beispielanwendung. 51

Abb. 4.1 Architektur des Modellsystems A222 (aus /PIL 18/). 53

Abb. 4.2 Übersicht (Netzwerkplan) des Modellsystems A222 im Modul 1 (TXS) des AnTeS. 54

Abb. 4.3 Übersicht des Modellsystems A222 im Modul 2 (Simulink) des AnTeS. 55

Abb. 4.4 Leittechnische Funktion (Funktionsplan) in der Processing Unit 1 (PU1.A) des Modellsystems A222 im Modul 1 von AnTeS und Ansteuerung von Fehlern/Ausfällen. 55

Abb. 4.5 Leittechnische Funktion (Funktionsplan) in der Processing Unit 1 56

Abb. 4.8 Beispielhafte Generierung von Fehlerraten im Modul 2 des AnTeS (in diesem Beispiel der Fehlerraten FR1, …, FR5 mit jeweils $1 \cdot 10^{-7}$ h^{-1}). 61

Abb. 4.9 Für das Beispiel in Abb. 4.8 berechnete Fehlerraten (FR1, …, FR5) als Funktion der Simulationszeit. 62

Abb. 4.10 Simulink-Subsystem, dass im Modellsystem A222 sofortige Reparaturen durchführt. ... 63

Abb. 4.11 Sofortige Reparaturen von selbstmeldenden Fehlern (SF) der VU1 (Detailansicht aus Abb. 4.10). ... 63

Abb. 4.12 Simulink-Subsystem zur Durchführung von WKPen im Modellsystem A222 ... 66

Abb. 4.13 Post-Processing im Modell A222 des AnTeS für die externe Ausgabe ... 67

Abb. 4.14 Sensitivitätsanalyse hinsichtlich Reparaturzeiten für das Modellsystem A133 mit Hilfe des Moduls 2 des AnTeS ... 69

Abb. 4.15 Vergleich der Sensitivitätsanalysen für das Modellsystem A133 mit dem Modul 2 des AnTeS und im Vorgängervorhaben /PIL 18/ ... 69

Abb. 4.16 Vergleich der Ergebnisse für das Modellsystem A222 ... 70

Abb. 4.17 Unterschied zwischen den Modellsystemen A222 und A222mod in den Funktionsplänen der PUs. ... 71

Abb. 4.18 Vergleich der Ergebnisse für das Modellsystem A222mod. ... 72

Abb. 4.19 Die begrenzte „Auflösung“ digitaler Zahlen kann zu systematischen Abweichungen führen. ... 74

Abb. 4.20 Fehlerraten (Vergleichswert sollte $1 \cdot 10^{-9}$ h^{-1} ergeben) in fünf unterschiedlichen Simulationsdurchläufen (FR1, … FR5). ... 75

Abb. 4.21 Mit SimGen erstellte generische Simulationen einer BE-Beckenkühlung und eines Forschungsreaktors. ... 78

Abb. 4.22 Mit SimGen erstellte Simulation von TeSys (Testsystem der GRS, siehe Abschnitt 2.3.4). ... 79

Abb. 4.23 Dynamischer Funktionsplan für den Strang 1 der BE-Beckenkühlung im graphischen Servicemonitor (GSM) auf dem Servicerechner des TXS ... 80

Abb. 4.24 Steuerung und Einstellungen von automatischen Analysen in KranSim ... 82

Abb. 4.25 Beispielergebnis einer allgemeinen Analyse des Bewegungsraums mit KranSim ... 83

Abb. 4.26 Ergebnis einer Lastabsturzanalyse ... 84

Abb. 4.27 Beispielergebnis für eine Analyse entlang eines definierten Pfads in KranSim. ... 87

Abb. A 1 Der Funktionsblock RTE-Input zur Erteilung bestimmter Freigaben (s. Text) 103

Abb. A 5 Auszug aus dem „Code Generation Report“ von Simulink für das betrachtete Modellsystem (unter „Code Interface Report“)..... 112

Abb. A 7 Grundlegender Aufbau der modellierten digitalen Leittechniksysteme ... 116

Abb. A 8 Modellsystem A222. 119

Abb. A 9 Modellsystem A120. 121

Abb. A 11 Die berechnete Verfügbarkeit (blau) und Unverfügbarkeit (rot) für das Modellsystem A120 als Funktion der Zeit (in Stunden) 124

Abb. A 13 Simulationssteuerung – Hinzufügen/Entfernen im Bearbeitungsmodus..... 128

Abb. A 22 Kranbewegungen mit unterschiedlichen Geschwindigkeiten..... 143

8 Abkürzungen

A/D	Analog-Digital(-Wandlung)
AI	Analog In
AL	Analog Logic
AnTeS	Analyse- und Testsystem
AO	Analog Out
AU	Acquisition Unit
BE	Brennelement
BELT	Betriebliche Leittechnik
BR	Bewegungsraum
D/A	Digital-Analog(-Wandlung)
DI	Digital In
DIMAS	Diagnosis and Maintenance Server
DLL	Direct Link Library
DLT	Digitale Leittechnik
DO	Digital Out
EEPROM	Electrically Erasable Programmable Read-Only Memory
FMEA	Failure Mode and Effects Analysis
GRS	Gesellschaft für Anlagen- und Reaktorsicherheit
GSM	Graphischer Servicemonitor
KKK	Kernkraftwerk Krümmel
KranSim	Kransimulation (bei der GRS entwickelte Software)
NSF	Nicht-selbstmeldender Fehler
PSA	Probabilistische Sicherheitsanalyse
PU	Processing Unit
RTE	Real Time Environment
SDI	TXS Digital Input Module
SDO	TXS Digital Output Module
SF	Selbstmeldender Fehler
SILT	Sicherheitsleittechnik
SimGen	Simulation Generator (bei der GRS entwickelte Software)
SPACE	Specification and Coding Environment (TXS)
SQL	Structured Query Language („Strukturierte Abfrage-Sprache“)
TeSys	Testsystem (bestehend aus verfahrenstechnischen Komponenten)
TXS	Teleperm XS (von Framatome, ehemals Areva)

VU	Voting Unit
V&V	Verifizierung und Validierung
WKP	Wiederkehrende Prüfung

A Anhang

A.1 Direkter Zugriff auf das Modul 1 von AnTeS (TXS) im laufenden Betrieb

Die Programmierung und auch die Überwachung des Moduls 1 des AnTeS erfolgt über den Servicerechner, der über eine Ethernet-Verbindung mit dem Leittechniksystem verbunden ist (siehe z. B. Abb. 2.4 auf Seite 3). Auf dem Servicerechner stehen, neben der Engineering-Umgebung SPACE zum Erstellen von leittechnischen Funktionen (Funktionsplänen), zusätzlich eine Reihe unterschiedlicher Werkzeuge zur Verfügung. So kann beispielsweise der sogenannte Graphische Servicemonitor (GSM) im laufenden Betrieb des Leittechniksystems dynamische Funktionspläne darstellen.

Weitere Werkzeuge auf dem Servicerechner sind eine Vielzahl von Servicetools zum Eingreifen und Überwachen des Systems. Als Kommunikationsschnittstelle zwischen den Servicetools und dem TXS dient der Diagnosis and Maintenance Server DIMAS. Für DIMAS steht eine Programmierschnittstelle (API - Application Programming Interface) zur Verfügung, über welche die Servicetools als Clients angebunden sind. DIMAS selbst kann aber auch bidirektional mit dem eigentlichen Leittechniksystem kommunizieren, weswegen DIMAS als Serviceschnittstelle eine zentrale Rolle einnimmt.

DIMAS erlaubt dem Benutzer über eine eigene Python-Umgebung auch den direkten Zugriff auf die Leittechnik (über die sogenannte DIMAS-Shell). Auf diese Weise können Funktionsplanvariablen, wie verarbeitete Signale, Funktionsblockparameter und Speicher aus der Leittechnik ausgelesen werden. Es können aber auch Werte, wie änderbare Parameter, Speicher oder Signalwerte, in die Leittechnik geschrieben werden, wenn die entsprechenden Freigaben vorliegen. Über die Programmierschnittstelle (API) von DIMAS kann dies auch über selbst erstellte Python-Skripte (Programme) erfolgen.

Als Schutz vor unautorisiertem Zugriff auf das Leittechniksystem über den Servicerechner des TXS wurde ein Zugriffsschutz auf mehreren Ebenen implementiert:

1. Der Passwortschutz auf Betriebssystemebene stellt sicher, dass nur authentifizierte Personen an der Wartungseinheit arbeiten können.
2. Um die Benutzer von Servicetools über die Verbindung zum DIMAS zu authentifizieren, werden für jeden Benutzer X.509 Private-Key/Zertifikatspaare generiert und von DIMAS während des Verbindungsaufbaus verifiziert.
3. Auf der Seite der Leittechnik werden bestimmte Befehle nur dann ausgeführt, wenn sich die Prozessormodule (SVE2) des Systems im dafür erforderlichen Betriebsmodus befinden und damit weitere Berechtigungsfreigaben erteilt sind.

Betriebsmodi der Prozessormodule des TXS

Unmittelbar nach dem Starten eines Prozessormoduls befindet sich dieses im Betriebsmodus STARTUP und wechselt üblicherweise nach dem Abschluss des Hochfahrvorgangs in den Betriebsmodus OPERATION. Darüber hinaus existieren aber auch noch weitere Betriebsmodi (TEST, DIAGNOSIS), in denen weitreichendere Eingriffe in das Leittechniksystem möglich sind. Standardmäßig ist der Wechsel in die Betriebsmodi TEST und DIAGNOSE allerdings nicht möglich.

Damit die Betriebsmodi TEST und DIAGNOSIS eines Prozessormoduls für den Benutzer zugänglich sind, muss sich auf einem Funktionsplan in der aktuell laufenden Software des Prozessormoduls ein Funktionsblock „RTE-INPUT“ befinden und dessen Eingänge entsprechend beschaltet sein (Abb. A 1). Über diesen können dann (u. a.) die Freigaben zum Wechsel in die Betriebsmodi TEST und DIAGNOSIS erteilt werden, in dem an die entsprechenden Eingänge („Enable FUNCTIONAL TEST“ bzw. „Enable DIAGNOSIS“) jeweils eine binäre 1 (für „True“) angelegt wird (in Bild durch feste Binärwertbausteine, es wären aber beispielsweise auch von außen eingespeiste Binärwerte über Schlüsselschalter o. ä. möglich).

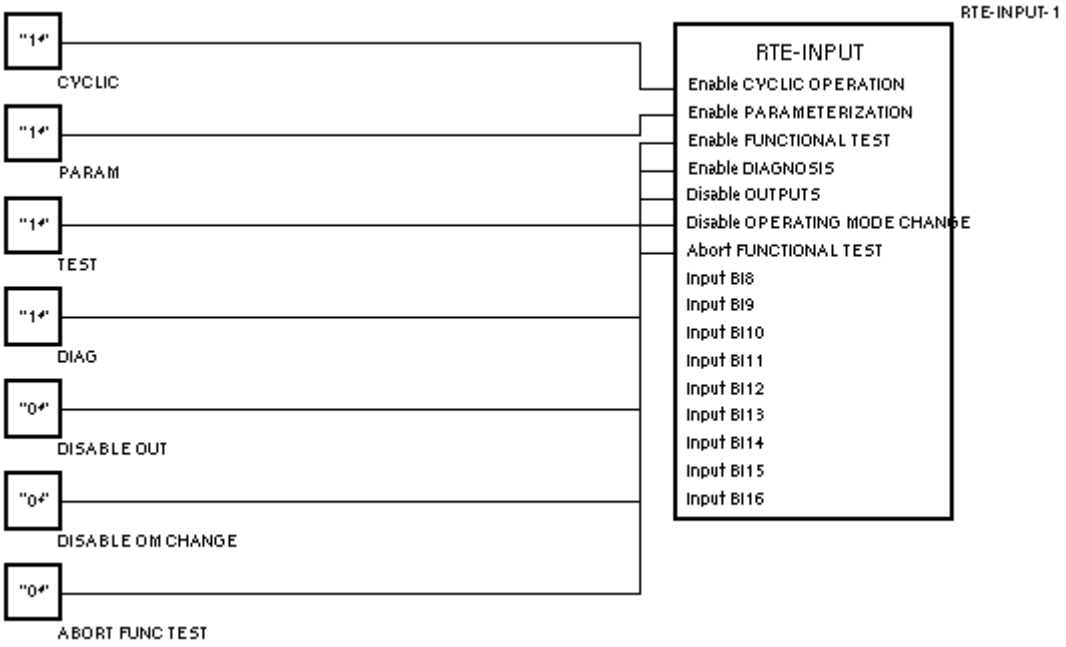


Abb. A 1 Der Funktionsblock RTE-Input zur Erteilung bestimmter Freigaben (s. Text)

Die einzelnen Betriebsmodi der Prozessormodule (SVE2) und die jeweils möglichen Eingriffe ins TXS sind:

- Betriebsmodus STARTUP:
  - Dies ist der Betriebsmodus beim Hochfahren der Prozessormodule, nachdem die eingebauten Selbsttests abgeschlossen sind und die Laufzeitumgebung die Kontrolle übernommen hat.
  - Erlaubte Zugriffe/Maßnahmen:
    - Lesen von Funktionsplan-Variablen (Parameter, Speicher und Signale)
    - Verifizierung der geladenen Software (über Prüfsummen oder byteweisen Vergleich)
- Betriebsmodus OPERATION
  - Dies ist der Standardbetriebsmodus der Prozessormodule, in dem diese zyklisch Funktionspläne abarbeiteten, auf Ein- und Ausgabemodule zugreifen, Signaldaten an andere CPUs senden und von ihnen empfangen.

- Erlaubte Zugriffe/Maßnahmen:
  - Lesen von Funktionsplan-Variablen (Parameter, Speicher und Signale)
  - Verifizierung der geladenen Software (über Prüfsummen oder byteweisen Vergleich)
- Liegen die entsprechenden Freigaben vor (über einen RTE-INPUT-Funktionsblock, s. o.), sind ggfs. zusätzlich folgende Zugriffe/Maßnahmen möglich:
  - Umschaltung in den Betriebsmodus TEST
  - Umschaltung in den Betriebsmodus DIAGNOSIS
  - Ändern von (in Funktionsplänen als änderbar gekennzeichneten) Parametern und Speichern von Funktionsblöcken während der Laufzeit
  - Speichern von (in Funktionsplänen als änderbar gekennzeichneten) Parametern im festen Speicher (EEPROM) der Prozessormodule, so dass diese Parameter auch nach einem Neustart aktiv bleiben
  - Wiederherstellen von änderbaren Parametern aus dem festen Speicher der Prozessormodule

- Betriebsmodus TEST:
  - Dies ist der Betriebsmodus zum Durchführen von Funktionstests.
  - Erlaubte Zugriffe/Maßnahmen:
    - Lesen und Ändern von Funktionsplan-Variablen (Parameter, Speicher und Signale)
    - Verifizierung der geladenen Software (über Prüfsummen oder byteweisen Vergleich)
    - Neustart oder Umschaltung auf Einzelschrittbearbeitung von Funktionsplänen
    - Wiederherstellen von veränderbaren Parametern aus dem festen Speicher der Prozessormodule
    - (Remote-)Neustart des Prozessormoduls

- Liegt die entsprechende Freigabe vor (über einen RTE-INPUT-Funktionsblock, s. o.), ist ggfs. zusätzlich folgende Maßnahme möglich:
  - Umschaltung in den Betriebsmodus DIAGNOSIS

- Betriebsart DIAGNOSIS:
  - In diesem Betriebsmodus werden alle Zugriffe/Maßnahmen der anderen Betriebsmodi ohne Einschränkungen unterstützt.
  - Zusätzlich ist auch die Aktualisierung der Anwendungssoftware auf dem entsprechenden Prozessormodul über das Netzwerk möglich („Remote-Hochladen von Funktionsplänen“)

Anmerkung: Ohne die entsprechende Freigabe (über RTE-Funktionsblöcke, s. o., sowie physische Kippschalter auf den entsprechenden Karten) kann über die Ethernet-Anbindung des Servicerechners an das Leittechniksystem keine neue Software auf die Prozessormodule aufgespielt werden, was aus sicherheitstechnischer Sicht natürlich deutlich günstiger ist. Software kann dann nur über eine direkte serielle Verbindung des Servicerechners (oder anderen Rechners/Laptops) an das jeweilige Prozessormodul erfolgen. Hierzu muss ein entsprechendes Kabel direkt mit einem dafür vorgesehenen Eingang auf der Frontseite des Prozessormoduls verbunden werden.

Fehlerinjektion mit Hilfe von DIMAS

Im laufenden Projekt wurden Fehlerkombinationen, z. B. für automatisierte FMEAs, durch Fehlerinjektion über die digitalen Eingangskarten (SDI) des TXS realisiert (siehe Abschnitt 3.1). Mit Hilfe von DIMAS lässt sich eine Fehlerinjektion eleganter und vor allem auch deutlich schneller in der Ausführung realisieren.

Über die API von DIMAS können eigene Python-Skripte (Programme) direkt auf dafür vorgesehene Parameter innerhalb des TXS zugreifen. Zukünftig werden daher in eigens dafür vorgesehenen Funktionsplänen für jeden anzusteuernden Fehler (der Fehlerinjektion) eigene Binärwertbausteine implementiert und als änderbar gekennzeichnet. Da sehr leicht sehr viele Binärwertbausteine in einem Funktionsplan untergebracht und angesteuert werden können, ist dann auch keine vergleichsweise komplizierte Fehleradressierung wie im Abschnitt 3.1 mehr notwendig. Vor allem können in einem einzigen (Berechnungs-)Zyklus des TXS sämtliche Fehler einer gewünschten Fehlerkombination gleichzeitig eingestellt werden.

Es ist zu erwarten, dass dann bei der Fehlerinjektion und insbesondere bei der Durchführung von automatisierten FMEAs mit dem Modul 1 von AnTeS (TXS) annähernd dieselbe Geschwindigkeit wie im Modul 2 von AnTeS erreicht werden kann.

A.2 Erstellung und Verwendung von DLLs im Modul 2 von AnTeS

Die Erzeugung dynamischer Programmbibliotheken aus Simulink-Modellen wird in diesem Abschnitt anhand eines sehr einfachen Beispiels erläutert.

Ausgangspunkt ist das Simulink-Modell in Abb. A 2. Diese Abbildung zeigt einen einfachen Funktionsplan mit einem Ein- und einem Ausgangssignal (welche sich jeweils aus dem Signalwert und dessen Validität zusammensetzen – vergleiche hierzu Abschnitt 2.2). Das Eingangssignal wird in einen Grenzwertbaustein („comp max“) geleitet, dort mit einem eingestellten Grenzwert („Threshold“) verglichen und anschließend das Ergebnis dieses Vergleichs ausgegeben.

Anm.: Der in Abb. A 2 gezeigte Grenzwertbaustein („comp max“) ist ein Simulink-Subsystem, dessen Verhalten dem entsprechenden TXS-Funktionsbaustein entspricht. Dieser wurde auf die in Abschnitt 2.2 beschriebene Weise erstellt.

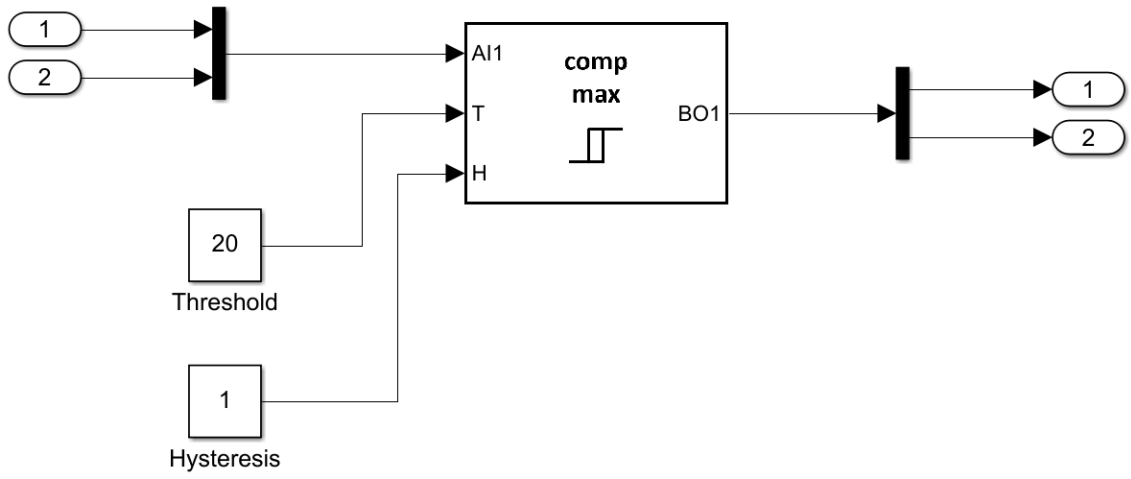


Abb. A 2 Funktionsplan, der nur einen Grenzwertbaustein („comp max“) enthält, für die Erläuterung der Erzeugung und Verwendung von DLLs.

Die Erstellung der entsprechenden DLL kann unmittelbar in Simulink durchgeführt werden. Wesentlich hierfür ist lediglich, dass einige Parameter der Konfiguration des Simulink-Modells korrekt eingestellt werden (siehe Abb. A 3 und Abb. A 4). Die Bedeutung der dargestellten Parameter kann der Dokumentation von Simulink entnommen werden.

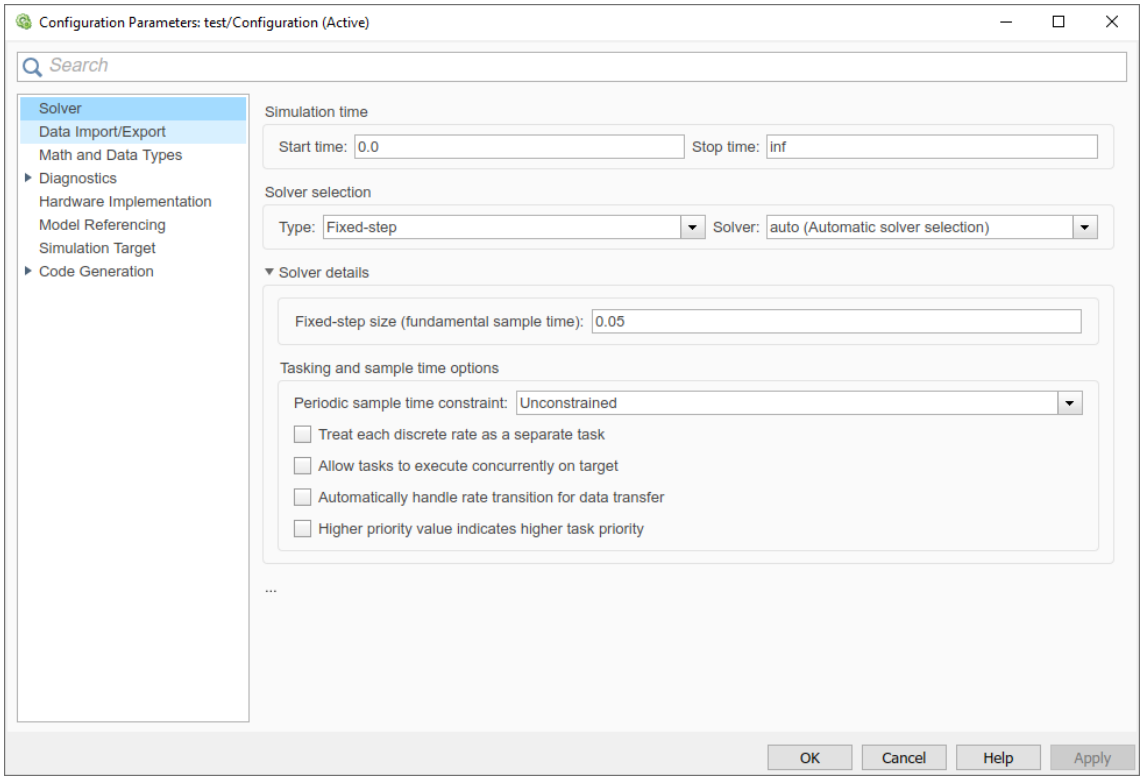


Abb. A 3 Wichtige Basiseinstellungen (für den „Solver“) in Simulink für die Erstellung von DLLs

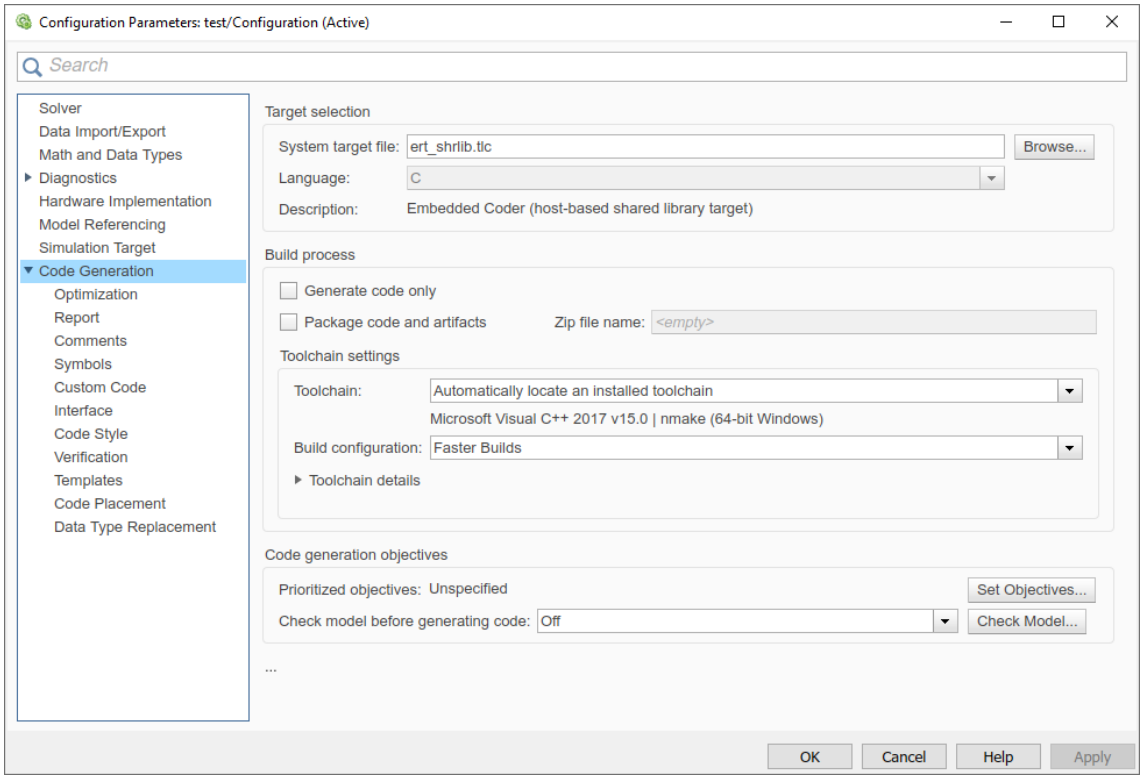


Abb. 4 Weitere notwendige Einstellungen (unter „Code Generation“) in Simulink für die Erstellung von DLLs

Nachfolgend wird beispielhaft dargestellt, wie der Funktionsplan aus Abb. A 2 verwendet werden kann, nachdem dieser in Simulink in eine DLL umgewandelt wurde. Zum Nachvollziehen sind Kenntnisse zu Python oder zumindest allgemeine Kenntnisse zur Programmierung hilfreich.

Das nachfolgende Python-Skript verwendet die dynamische Programmbibliothek (DLL), die aus dem Funktionsplan aus Abb. A 2 erzeugt wurde (Erläuterungen folgen nach dem Skript):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ********************************
# *  Module2_Demo ****************
# *  Author: C. Mueller, GRS *******
# *  File: Module2_Demo.py *********
# *  Date: 29.09.2020 **************
# ********************************


# ********************************
# Imports ************************
# ********************************
from ctypes import *
```

```
import os
import tkinter
from tkinter import *


# *********************************
# Classes ***********************
# *********************************

# Struct Inputs
class Inputs(Structure):
    _fields_ = [("In1", c_double),
                ("In2", c_double)]

# Struct Outputs
class Outputs(Structure):
    _fields_ = [("Out1", c_bool),
                ("Out2", c_bool)]


# *********************************
# Initialize Model ****************
# *********************************

# Load dll File
actualDirectory = os.path.dirname(os.path.abspath(sys.argv[0]))
actualFile = os.path.join(actualDirectory, "Module2_Demo_win64.dll")
dllLoaded = cdll.LoadLibrary(actualFile)

# Call Initialize
dllLoaded.Module2_Demo_initialize()

# Calc One Step (to be on the safe side)
dllLoaded.Module2_Demo_step()

# In- and Outputs
input = Inputs.in_dll(dllLoaded, 'Module2_Demo_U')
output = Outputs.in_dll(dllLoaded, 'Module2_Demo_Y')

# Set both Inputs to "0"
input.In1 = float(0)
input.In2 = float(0)
dllLoaded.Module2_Demo_step()


# *********************************
# Main Window *********************
# *********************************

# Create Window
root = tkinter.Tk()
root.title("AnTeS Module 2 Demo")
widthWindow = 1024
heightWindow = 576

# Center Window
x = (root.winfo_screenwidth() // 2) - (widthWindow // 2)
y = (root.winfo_screenheight() // 2) - (heightWindow // 2)
root.geometry("{}x{}+{}+{}".format(widthWindow, heightWindow, x, y))

# Load Images
```

```
actualFile = os.path.join(actualDirectory, "background_2.gif")
imageBackground = PhotoImage(file=actualFile)

# Background Frame
frameMain = Frame(root, bd=0, width=widthWindow, height=heightWindow)
frameMain.pack()

# Background
labelBackground = Label(frameMain, image=imageBackground)
labelBackground.place(x=0, y=0, relwidth=1, relheight=1)

# Function for Scale und Radiobuttons
def sel(event):
    selection_up = var_up.get()
    selection_down = switch_variable.get()
    label_up.config(text=str(selection_up))
    label_down.config(text=str(selection_down))
    input.In1 = float(var_up.get())
    if switch_variable.get():
        input.In2 = float(2)
    else:
        input.In2 = float(0)
    dllLoaded.Module2_Demo_step()
    label_up_out.config(text=str(output.Out1))
    label_down_out.config(text=str(output.Out2))

# Help-Function to call sel() without argument
def sel2():
    sel(1)

# Variable for Scale
var_up = DoubleVar()

# Scale
scale_up = Scale(root, variable=var_up, orient="horizontal",
                 resolution=0.1, from_=15, to=25, command=sel,
                 showvalue=0, sliderlength=30, width=25, length=180)
scale_up.place(x=40, y=218)

# Label for Scale
label_up = Label(root)
label_up.config(font=("Microsoft Sans Serif", 20, "bold"))
label_up.place(x=110, y=280)

# Label for Radiobutton
label_down = Label(root)
label_down.config(font=("Microsoft Sans Serif", 20, "bold"))
label_down.place(x=110, y=345)

# Label for Scale
label_up_out = Label(root)
label_up_out.config(font=("Microsoft Sans Serif", 20, "bold"))
label_up_out.config(text="1")
label_up_out.place(x=840, y=280)

# Label for Radiobutton
label_down_out = Label(root)
label_down_out.config(font=("Microsoft Sans Serif", 20, "bold"))
label_down_out.config(text="0")
label_down_out.place(x=840, y=345)
```

```
# Additional two Labels
label_temp = Label(root)
label_temp.config(font=("Microsoft Sans Serif", 20, "bold"))
label_temp.config(text="Threshold: 20")
label_temp.place(x=760, y=450)
label_temp = Label(root)
label_temp.config(font=("Microsoft Sans Serif", 20, "bold"))
label_temp.config(text="Hysteresis: 1")
label_temp.place(x=760, y=515)

# Button for Error Flag
switch_variable = tkinter.BooleanVar(value=False)
toggle_button1 = Radiobutton(root, text="Error ON",
                             variable=switch_variable,
                            indicatoron=False, value=True,
                             width=8, command=sel2)
toggle_button1.config(font=("Microsoft Sans Serif", 12, "bold"))
toggle_button1.place(x=135, y=413)

toggle_button2 = Radiobutton(root, text="Error OFF",
                             variable=switch_variable,
                             indicatoron=False, value=False,
                             width=8, command=sel2)
toggle_button2.config(font=("Microsoft Sans Serif", 12, "bold"))
toggle_button2.place(x=35, y=413)

# call sel() once to have values from the beginning
sel(1)

# ********************************
# Main Loop ************************
# ********************************
if __name__ == '__main__':
    root.mainloop()
```

Es folgt die ausführliche Erläuterung der einzelnen Zeilen/Programmblöcke:

```
from ctypes import *
import os
import tkinter
from tkinter import *
```

Hier werden Python-Module bzw. Funktionen aus diesen Modulen importiert. Das Modul „ctypes“ ist erforderlich, um in der korrekten Form und mit den richtigen Variablentypen mit der DLL zu kommunizieren (Ein- und Ausgabekanäle des Simulink-Modells). Das Modul „os“ ist für Interaktionen mit dem Filesystem notwendig, „tkinter“ wird für die Erstellung des Programmfensters benötigt.

Unmittelbar zu Beginn des Skriptes werden zwei Klassen definiert, die die Ein- und Ausgabekanäle der DLL beschreiben:

```
# Struct Inputs
class Inputs(Structure):
```

```
    _fields_ = [("In1", c_double),
                ("In2", c_double)]

# Struct Outputs
class Outputs(Structure):
    _fields_ = [("Out1", c_bool),
                ("Out2", c_bool)]
```

Die Namen und Variablentypen für die einzelnen Ein- und Ausgangskanäle, welche hier angegeben werden müssen, ergeben sich aus dem Code Generation Report in Simulink, der nach der Erzeugung der DLL angezeigt wird. Für dieses Beispiel sieht das im „Code Generation Report“ von Simulink unter dem Punkt „Code Interface Report“ wie in Abb. A 5) aus.

Inports

Block Name	Code Identifier	Data Type	Dimension
<Root>/In1	Module2_Demo_U.In1	real_T	1
<Root>/In2	Module2_Demo_U.In2	real_T	1

Outports

Block Name	Code Identifier	Data Type	Dimension
<Root>/Out1	Module2_Demo_Y.Out1	boolean_T	1
<Root>/Out2	Module2_Demo_Y.Out2	boolean_T	1

Abb. A 5 Auszug aus dem „Code Generation Report“ von Simulink für das betrachtete Modellsystem (unter „Code Interface Report“).

In Abb. A 5 lässt sich ablesen, dass es zwei Eingangskanäle („In1“ und „In2“) und zwei Ausgangskanäle („Out1“ und „Out2“) gibt sowie dass deren Datentypen „real_T“ bzw. „boolean_T“ sind. Wie die angegebenen Datentypen zu verstehen sind, kann in der Simulink-Hilfe nachgelesen werden. Für diesen Fall sind die entsprechenden Datentypen im Python-Skript „c_double“ und „c_bool“. Um diese verwenden zu können, musste zuvor das Modul „ctypes“ importiert werden (s.o.).

Der nachfolgende Abschnitt des Skripts lädt die DLL und initialisiert das Modell:

```
# Load dll File
actualDirectory = os.path.dirname(os.path.abspath(sys.argv[0]))
actualFile = os.path.join(actualDirectory, "Module2_Demo_win64.dll")
dllLoaded = cdll.LoadLibrary(actualFile)

# Call Initialize
dllLoaded.Module2_Demo_initialize()

# Calc One Step (to be on the safe side)
dllLoaded.Module2_Demo_step()

# In- and Outputs
input = Inputs.in_dll(dllLoaded, 'Module2_Demo_U')
```

```
output = Outputs.in_dll(dllLoaded, 'Module2_Demo_Y')

# Set both Inputs to "0"
input.In1 = float(0)
input.In2 = float(0)
dllLoaded.Module2_Demo_step()
```

Die entsprechenden Methoden (und deren Namen) können ebenfalls dem „Code Generation Report“ in Simulink nach der Erzeugung der jeweiligen DLL entnommen werden. In diesem Beispiel muss das Simulink-Modell zunächst über den Aufruf der Methode „[...]_initialize()“ für die Verwendung vorbereitet werden. Im Einzelnen werden hier nacheinander also die folgenden Schritte durchgeführt:

1. Lade die DLL-Datei.
    a. Die entsprechende Instanz der geladenen Datei heißt hier „dllLoaded“.
2. Initialisiere das Modell (in der entsprechenden Instanz).
    a. „dllLoaded.Module2_Demo_initialize()“.
3. Führe einen einzelnen Berechnungsschritt aus.
    a. „dllLoaded.Module2_Demo_step()“.
4. Aktualisiere die Ein- und Ausgangswerte:
    a. „input = Inputs.in_dll(dllLoaded, ‘Module2_Demo_U‘)“,
    b. „output = Outputs.in_dll(dllLoaded, ‘Module2_Demo_Y‘)“.
    c. Dies ist die Anwendung von Methoden auf die Instanzen der zuvor definierten Ein- und Ausgabeklassen „Inputs“ und „Outputs“.
5. Setze die Eingangswerte für „In1“ und „In2“ jeweils auf „0“:
    a. „input.In1 = float(0)“,
    b. „input.In2 = float(0)“.
    c. Die entsprechenden Eingangsnamen „In1“ und „In2“ wurden zuvor in der Klasse „Inputs“ definiert, „input“ ist eine Instanz dieser Klasse.
6. Führe einen weiteren Berechnungsschritt aus:
    a. „dllLoaded.Module2_Demo_step()“.

Anschließend folgt im Code die Definition des Hauptfensters („Main Window“) mit Hilfe des Python-Moduls „tkinter“. Schlussendlich wird ganz am Ende des Skriptes die eigentliche Programmausführung gestartet („root.mainloop()“ – Details zur Handhabung sind im Internet zu finden). Innerhalb des Hauptfensters wird bei jeder Veränderung der Eingangsparameter (z. B. über Schalter oder Schiebregler) eine Funktion („sel(event)“) aufgerufen, die die Ausgangswerte der DLL ausliest und anschließend den nächsten Berechnungsschritt anstößt.

Führt man dieses Python-Skript aus (vorausgesetzt die entsprechende DLL und die zu importierenden Bilder befinden sich mit im selben Verzeichnis), so stellt sich dieses Programm wie in Abb. A 6 gezeigt dar. Über einen Schieberegler kann der Eingangswert dynamisch verändert werden. Das ERROR-Attribut des Eingangssignals kann über die Schalter „Error On“ und „Error Off“ gesetzt bzw. gelöscht werden. Auf der rechten Seite wird angezeigt, ob der Grenzwert (unter Berücksichtigung der Hysterese) überschritten ist oder nicht. Zusätzlich wird auch das ERROR-Attribut des Ausgangssignals mit angezeigt.

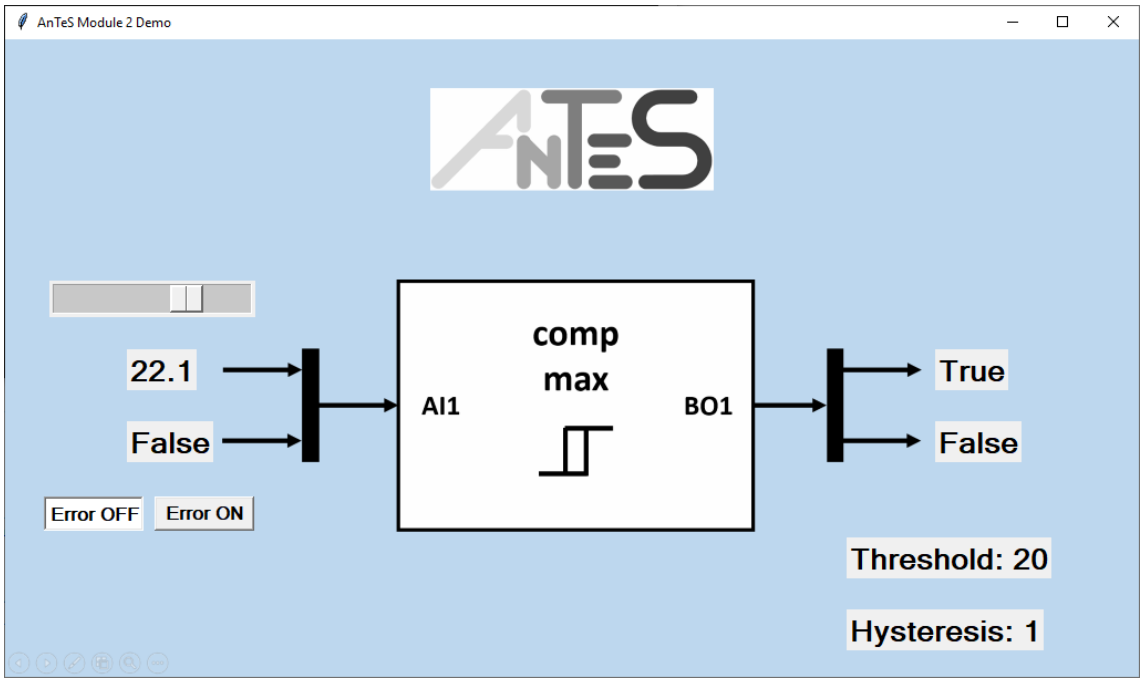


Abb. A 6 Einfaches Programm zur Demonstration der Benutzung eines Grenzwertbausteins („comp max“). Die zugrundeliegende Logik des entsprechenden Funktionsbausteins liegt hierbei als DLL vor, die mit Simulink erstellt wurde.

A.3 Modellsysteme und Analysemethodik aus dem Vorgängervorhaben

In diesem Abschnitt werden zunächst die Modellsysteme aus dem Vorgängervorhaben /PIL 18/ beschrieben, da diese im aktuellen Vorhaben für Tests verwendet wurden. Diese Tests wurden durchgeführt, um einerseits die in diesem Vorhaben entwickelten neuen Methoden zu testen, aber auch um die im Vorgängervorhaben entwickelte Analysemethodik zu validieren. Diese wird daher ebenfalls in diesem Abschnitt erläutert.

Modellsysteme

Grundsätzlich bestehen die Modellsysteme aus den Funktionseinheiten eines rechnerbasierten Leittechniksystems. Diese berücksichtigen konsistent sowohl Architekturaspekte (z. B. Kommunikationswege) als auch die Funktionsweise der vernetzten Hardware digitaler Leittechnik (u. a. Analogsignalverarbeitung, logische Verknüpfungen, Antriebssteuerung).

Alle potentiellen Ausfälle (Ausfallarten) von Funktionseinheiten wurden entsprechend der Annahmen aus den Analysen /FRW 06/, /HEJ 15/ und Vorgaben aus /NEA 15/ in zwei Gruppen unterteilt:

- erkannte Ausfälle (SF, engl. Self-signaling Failure) oder
- nichterkannte Ausfälle (NSF, engl. Non Self-signaling Failure).

Die Signalverarbeitung der modellierten digitalen Leittechnikarchitekturen findet auf drei Ebenen statt (Abb. A 7), wobei jede Ebene aus mehreren redundanten Rechnern bestehen kann, z. B.

- Erfassungsebene AU (engl. Acquisition Unit): Rechner AU1, AU2, AU3 …,
- Verarbeitungsebene PU (engl. Processing Unit): Rechner PU1, PU2, PU3 …,
- Steuerungsebene VU (engl. Voting Unit): Rechner VU1, VU2, VU3 ….

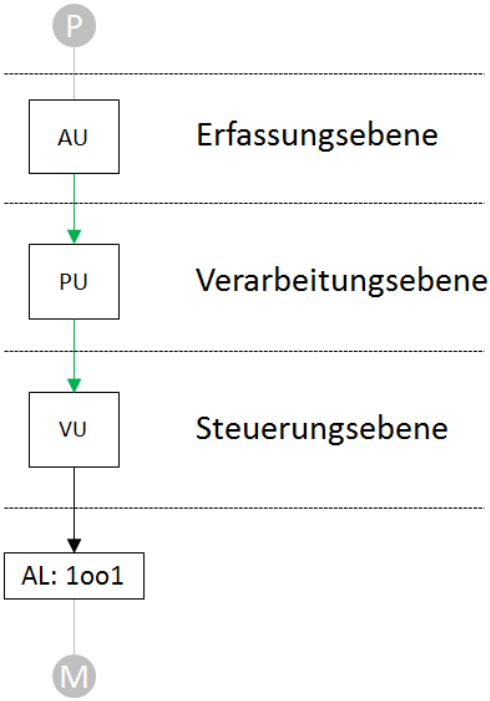


Abb. A 7 Grundlegender Aufbau der modellierten digitalen Leittechniksysteme

Architekturen von Leittechniksystemen gemäß Abb. A 9 können für die Modellsysteme des Vorgängervorhabens und dieses Vorhabens wie folgt beschrieben werden:

- Erfassungsebene (Messwerterfassung), bestehend aus AU-Rechnern:
  - In der Erfassungsebene werden die Messwerte (Prozessparameter der Anlage, wie z. B. Druck, Temperatur, Füllstand, Durchsatz) in redundanten Erfassungsrechnern (AU) erfasst, digitalisiert und als Datentelegramme über das Kommunikationsnetzwerk an die nächste Ebene weitergeleitet.
  - Erkannte Ausfälle werden mit einem entsprechenden Wert (engl. Flag) (Ausfall: „1“, valides Signal: „0“) markiert. Wurde der Fehler erkannt und das Flag auf „1“ gesetzt, so handelt es sich um einen selbstmeldenden Fehler (SF). Ist das Flag hingegen auf „0“ gesetzt, kann es sich sowohl um ein wahres Signal (OK) als auch um ein unerkannt fehlerhaftes Signal (NSF) handeln.
- Verarbeitungsebene (logische Signalverarbeitung), bestehend aus PU-Rechnern:
  - In den redundanten Verarbeitungsrechnern (PU) werden die auf der Erfassungsebene aufbereiteten Messwerte der Prozessparameter durch die Anwendersoftware (LEFU, „Leittechnik-Funktion“) verarbeitet.

- Die validen Werte aus den AUs (Flag „0“) werden in aufsteigender Reihenfolge sortiert und daraus das zweite Maximum (2. Max) bestimmt. Ist eines der Eingangssignale des Verarbeitungsrechners selbstmeldend ausgefallen (Flag „1“), wird das zweite Maximum nur aus den restlichen Signalen des zu überwachenden Parameters gebildet. Sind alle Eingangssignale des Verarbeitungsrechners bis auf eines selbstmeldend ausgefallen, wird das verbliebene Eingangssignal direkt für die mögliche Bildung eines Anregesignals (bei Überschreitung des entsprechenden Grenzwerts) verwendet (Max).

- Das zweite Maximum (oder Maximum, s. o.) wird mit einem Grenzwert verglichen und gegebenenfalls ein binäres Steuersignal („1“ – „EIN“ bzw. „0“ – „AUS“) zur Weiterleitung an die nächste Ebene generiert. Auch hier werden die erkannten Ausfälle der Rechner (PU) bzw. der Kommunikation zwischen den PUs und VUs mit einem Flag „1“ markiert und daher durch die Signalverarbeitung in den VU als ungültige Signale erkannt.

- Steuerungsebene (2-von-3-Auswahllogik), bestehend aus VU-Rechnern:

  - In den redundanten Voting Units (VU) werden die binären Signale der vorherigen Ebene durch eine 2-von-3-Auswahl ausgewertet und gegebenenfalls binäre Signale („1“ – EIN bzw. „0“ – „AUS“) zur Steuerung einer verfahrenstechnischen Komponente erzeugt. Liegen also an mindestens zwei der drei Eingänge des Voters valide Anregesignale (Signal „1“, Flag „0“) vor, so wird die entsprechende Leittechnikfunktion aktiviert.

  - Die Ausgangssignale der VUs enthalten keine Fehlererkennungsinformationen, selbstmeldende Fehler werden jedoch gemeldet und können dann gegebenenfalls repariert werden.

  - Fallen ein oder zwei der Eingangssignale selbstmeldend aus, wechselt der Voter intern zu einer 1-von-2-Auswahl, so dass bei einem oder zwei selbstmeldenden Ausfällen ein einzelnes valides Anregesignal für eine Auslösung ausreicht.

Die Antriebssteuerung einer verfahrenstechnischen Komponente des Systems in Abb. A.9 (M) wurde vereinfacht durch die analoge Logik (AL, z. B. Schaltanlage) modelliert.

Bei der Modellierung der Signalverarbeitung der generischen digitalen Leittechniksysteme werden darüber hinaus folgende konservative Annahmen getroffen:

- Die Kommunikation zwischen den Rechnern erfolgt über ein Netzwerk, welches für die Sicherheitsleittechnik typisch ist. Das heißt, es wird angenommen, dass alle Hardwareausfälle im Kommunikationsnetzwerk der Sicherheitsleittechnik immer erkannt werden und daher selbstmeldend sind. Vereinfachend werden daher die Ausfallraten der Kommunikationswege (beispielsweise zwischen AUs und PUs) direkt in den Ausfallraten der entsprechenden signalsendenden Komponenten berücksichtigt (z. B. selbstmeldende Ausfälle von AUs).
- Die Kommunikation zwischen den Ebenen erfolgt in vereinfachter Form, wobei die Signale der Erfassungsebene einen (digitalen) Zahlenwert und die Signale der Verarbeitungsebene binäre Werte darstellen. Hierbei wird zwischen dem Vorliegen eines Anregesignals („1“) und keines Anregesignals („0“) unterschieden. Außerdem wird die Übertragung jedes Signals mit einem Flag gekennzeichnet. Wird ein Signal als fehlerhaft erkannt, wird das Flag auf „1“ gesetzt. Es handelt sich dann um einen selbstmeldenden Fehler (SF). Ist das Flag auf „0“ gesetzt, kann es sich sowohl um ein wahres Signal (OK) als auch um ein unerkannt fehlerhaftes Signal (NSF) handeln.
- Nichtselbstmeldend ausgefallene AUs geben den minimalen Messwert aus.
- Nichtselbstmeldend ausgefallene PUs geben eine logische „0“ aus.
- Ausgefallene VUs geben eine logische „0“ aus.
- Die Software der AUs, PUs und VUs kann sowohl selbstmeldende als auch nichtselbstmeldende Ausfälle verursachen und wird nicht explizit modelliert. Die berücksichtigten Ausfallraten wurden nur über die möglichen Hardware-Ausfälle bestimmt.

Die Messeinrichtungen (u. a. Sensoren, Messumformer), Stromversorgung und Schnittstellen der Leittechnik werden in den vereinfachten Modellen nicht explizit berücksichtigt.

Die Modellbezeichnung der Modellsysteme ist als alphanumerische Darstellung der Leittechnikarchitektur aufgebaut:

- ein Buchstabe (z. B. A, B) für die unterschiedlichen Varianten der Leittechnikarchitektur (z. B. für jedes Teilsystem, das auf der Basis der Hardware und Software einer spezifischen Leittechniktechnologie aufgebaut ist),
- die Anzahl von redundanten Komponenten jeder Signalverarbeitungsebene einer Leittechnikarchitektur (z. B. Modelle: A222, A133, A333, A133B133).

Das Modellsystem A222 (Abb. A 8) besteht aus zwei VUs, zwei PUs und zwei AUs. Die zusätzliche Kennzeichnung der Modellsysteme mit Buchstaben (hier „A“) soll die Unterscheidung zwischen diversitären Teilsystemen ermöglichen. Die einzelnen Komponenten werden zusätzlich gemäß ihrer Redundanzzuordnung durchnummeriert. So bezeichnet beispielsweise AU1.A die AU der ersten Redundanz der Diversität A.

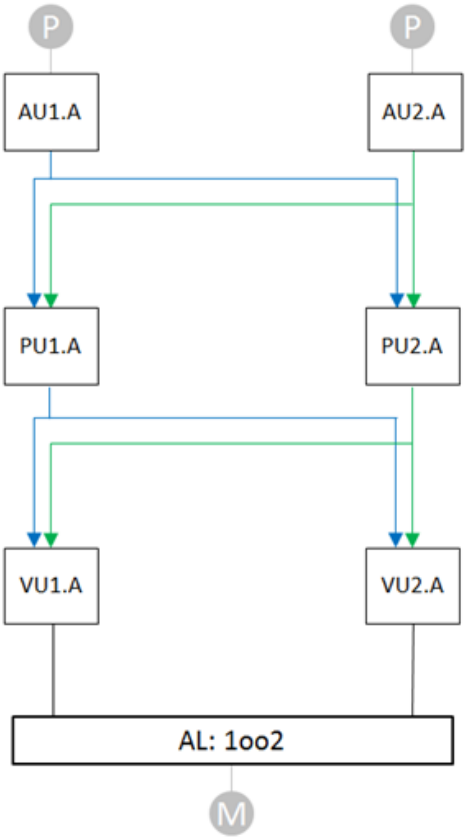


Abb. A 8 Modellsystem A222.

Da in der Regel die betrachteten Modellsysteme über mehr als eine VU verfügen, sind in allen Systemen den VUs jeweils zusätzlich analoge Baugruppen (AL) nachgeschaltet. Diese führen in Abhängigkeit der Anzahl der Eingangssignale (aus den vorgelagerten VUs) eine Wertung durch, z. B. für das Modellsystem A222 eine 1-von-2-Auswahl („1oo2“ - „1-out-of-2“) und für das Modellsystem A133 eine 1-von-1-Auswahl („1oo1“).

Die analogen Baugruppen können selbst Ursache von Ausfällen sein. In den Modellen werden im Hinblick auf spätere Modellerweiterungen die ALs mit der nicht-selbstmeldenden Ausfallart „Schaltet nicht“ (NSF) berücksichtigt.

Bei der Fehlerbaummodellierung der Ausfälle mittels der Software RiskSpectrum wird diese Ausfallart allerdings mit der willkürlich bestimmten sehr kleinen Ausfallrate von $1 \cdot 10^{-10}$ h^{-1} festgelegt, da nur Effekte innerhalb der digitalen Leittechnik betrachtet werden sollen.

Weitere Informationen zu den Modellsystemen, insbesondere z. B. zu Reparaturzeiten und Testzyklen sowie den Ausfallraten, können dem Bericht zum Vorgängervorhaben /PIL 18/ entnommen werden.

Analysemethodik

Für die übersichtliche und allgemeinverständliche Darstellung der Methoden aus dem Vorgängervorhaben /PIL 18/ wird im Folgenden das sehr vereinfachte Modellsystem A120 verwendet (Abb. A 9), welches nachfolgend kurz erläutert wird:

- Zwei Messungen (hier „P“ für Druckmessungen), bei denen angenommen wird, dass sie stets fehlerfrei funktionieren und ihre Signale fehlerfrei an die oberste Ebene der Leittechnik weitergeben, sind jeweils an eine Zusammenfassung von AU- und PU-Rechnern (engl. Acquisition and Processing Unit, APU) angeschlossen. Diese erfassen die Messwerte und überwachen die Eingangssignale auf Überschreitung eines MAX-Grenzwerts. Wird der Grenzwert überschritten, so gibt die jeweilige APU eine logische „1“ aus.
- Die Voting Unit VU1 bewertet die Eingangssignale mit einer 1-von-2-Auswahl. Stehen also ein oder zwei Signale mit einer logischen „1“ am Eingang der VU1 an, so gibt sie einen Startbefehl an den angeschlossenen Motor (M) aus. Der angeschlossene Motor soll stets fehlerfrei auf Signale von der VU1 reagieren.

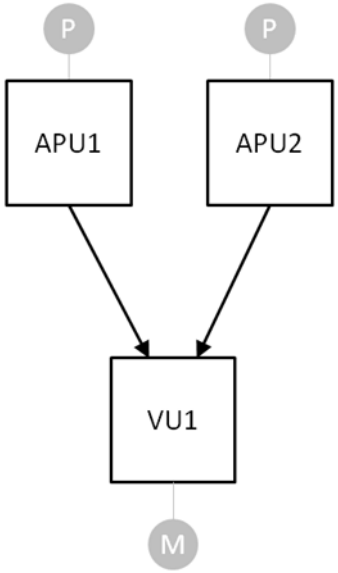


Abb. A 9 Modellsystem A120.

Es wird angenommen, dass alle Kommunikationswege stets fehlerfrei funktionieren. Die leittechnischen Komponenten können entweder fehlerfrei funktionieren (OK) oder nicht-selbstmeldend ausfallen (NSF). Für dieses Beispielmodell gibt es daher drei Basisereignisse (siehe Tab. A 1).

Tab. A 1 Basisereignisse des Beispielmodells A120 mit Ausfallraten.

Basisereignis	**Beschreibung**	**Ausfallrate λ**
APU1.NSF	Nicht-selbstmeldender Ausfall von APU1	$4 \cdot 10^{-2}\ h^{-1}$
APU2.NSF	Nicht-selbstmeldender Ausfall von APU2	$4 \cdot 10^{-2}\ h^{-1}$
VU1.NSF	Nicht-selbstmeldender Ausfall von VU1	$2 \cdot 10^{-2}\ h^{-1}$

Die folgende Tab. A 2 gibt eine Übersicht über alle denkbaren Zustände der Komponenten des Systems A120. Sie berücksichtigt alle Kombinationen von funktionsfähigen (OK) und ausgefallenen Komponenten (NSF). Der Gesamtzustand des Systems muss zu einem beliebigen Zeitpunkt einer Zeile dieser Tabelle entsprechen. Die letzte Spalte gibt an, ob das Modellsystem A120 aufgrund der jeweiligen Kombination als ausgefallen gelten muss (und damit unverfügbar ist) oder nicht.

Tab. A 2 Übersicht über die möglichen Komponentenzustände des Modells A120.

lfd. Nummer	APU1	APU2	VU1	Gesamtausfall
1	OK	OK	OK	nein
2	NSF	OK	OK	nein
3	OK	NSF	OK	nein
4	OK	OK	NSF	ja
5	NSF	NSF	OK	ja
6	NSF	OK	NSF	ja
7	OK	NSF	NSF	ja
8	NSF	NSF	NSF	ja

Die Fehlerbaumanalyse für das Modellsystem A120 wurde mit der Software RiskSpectrum /RIS 21/ durchgeführt. Da in den Analysen üblicherweise Test- und Reparaturzeiten berücksichtigt werden, wurden auch für dieses einfache Beispiel die entsprechenden Berechnungsmodelle der Fehlerbaumanalyse verwendet. Dabei wurden allerdings die Zeitintervalle für Test- und Reparaturzeiten so groß gewählt, dass innerhalb der hier betrachteten Zeiträume keine Prüfungen der Komponenten durchgeführt und daher Ausfälle nicht entdeckt oder behoben wurden.

Der Fehlerbaum für das Modellsystem A120 lässt sich in der Software RiskSpectrum wie in Abb. A 10 gezeigt darstellen.

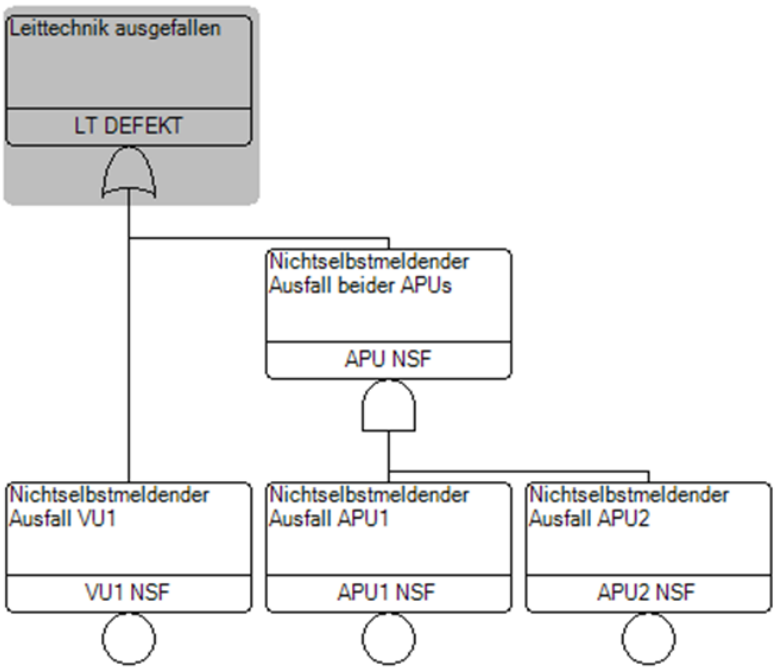


Abb. A 10 Fehlerbaum für das Modellsystem A120.

Wie im vorangegangenen Abschnitt wurde auch hier von den folgenden Ausfallraten ausgegangen:

- Ausfallrate einer APU $4 \cdot 10^{-2}$ h^{-1} für APU1 und APU2
- Ausfallrate einer VU1 $2 \cdot 10^{-2}$ h^{-1} für VU1

Die Rechnung mit der Analysesoftware RiskSpectrum liefert hierzu zeitabhängige Ergebnisse. In Abb. A 11 ist die Wahrscheinlichkeit für die Verfügbarkeit bzw. Unverfügbarkeit des Modellsystems A120 in Abhängigkeit von der Zeit für die ersten 100 Stunden dargestellt.

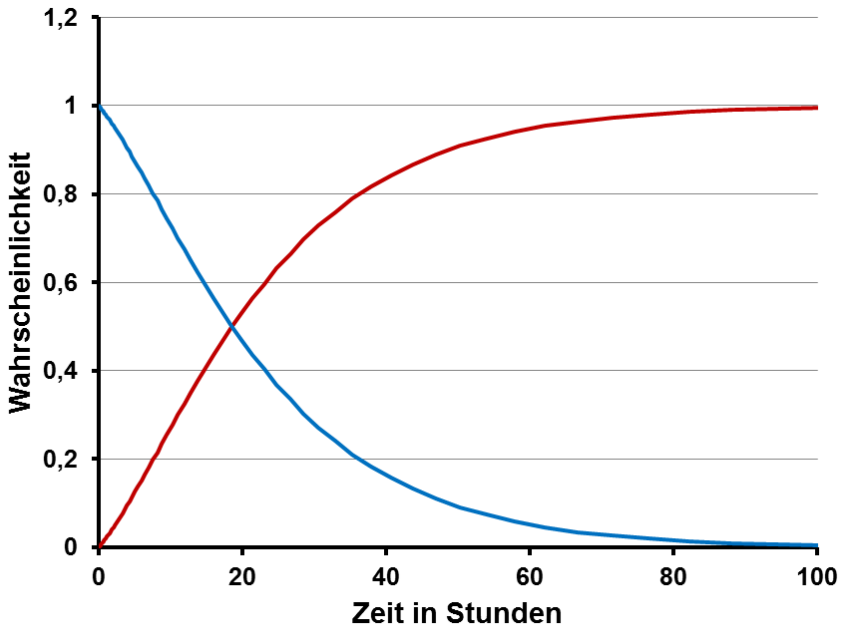


Abb. A 11 Die berechnete Verfügbarkeit (blau) und Unverfügbarkeit (rot) für das Modellsystem A120 als Funktion der Zeit (in Stunden)

A.4 Detaillierte Beschreibung von SimGen

SimGen ist derart programmiert, dass grundlegend nur zwischen zwei unterschiedlichen Typen von Elementen unterschieden wird: Nodes und Connections. Nodes sind Knotenpunkte, in welchen Wasser gespeichert wird und für die jeweils bestimmte Zustandsgrößen während der Simulation eines Systems gespeichert und aktualisiert werden. Nodes werden untereinander durch Connections gekoppelt, die im Wesentlichen Wasser zwischen diesen transportieren können.

Verfahrenstechnische Simulationen lassen sich in SimGen durch das Hinzufügen von Nodes und Connections über die obere Menüleiste aufbauen. Nach dem Hinzufügen eines Elements, sind dessen Eigenschaften (u. a. Startwerte für die Simulation) über einen Doppelklick auf das entsprechende Element in dafür vorgesehenen Eigenschaftsfenstern erreichbar. Die Positionierung der Elemente kann per Drag&Drop mit der Maus erfolgen oder wahlweise punktgenau über die entsprechenden Einstellwerte in den Eigenschaftsfenstern erfolgen.

Zum aktuellen Zeitpunkt stehen in SimGen die folgenden Elemente für die Erstellung von Simulationen über die obere Menüleiste zur Verfügung:

- Menü „Nodes“
  - Node
  - Node unendlich
- Menü „Behälter“ (ebenfalls Nodes)
  - Standard
  - BE-Becken
  - Pool-Reaktor
- Menü „Pumpen“ (Connections)
  - Orientierung nach rechts
  - Orientierung nach links
  - Orientierung nach oben
  - Orientierung nach unten
- Menü „Ventile“ (Connections)
  - Ventil horizontal
  - Ventil vertikal
- Menü „Wärmetauscher“ (Connections)
  - Wärmetauscher horizontal
  - Wärmetauscher vertikal
- Menü „Rohrleitungen“ (Connections)
  - Rohrleitung

Zusätzlich können über die Menüleiste auch noch Messungen und Teilsteuerungen platziert werden sowie die Eigenschaften der Anbindung an die Leittechnik (Menü „Interface“) erreicht werden.

Grundlegende Funktionen sind darüber hinaus über die Menüs „Datei“ und „Bearbeiten“ erreichbar. Die Software SimGen kann über den Menüpunkt „Beenden“ im Menü Datei beendet werden.

SimGen speichert alle Informationen zu Simulationen in einer eigenen SQLite-Datenbank, die als Datei im Installationsverzeichnis liegt. Änderungen an Simulationen können durch „Commit“ im Menü „Datei“ dauerhaft gespeichert werden. Bevor dies durchgeführt wurde, sind Änderungen jeweils nur vorläufig (und daher auch noch nicht dauerhaft gespeichert) und können im Menü „Datei“ über „Rollback“ zurückgenommen werden (in diesem Fall werden alle Änderungen seit dem letzten „Commit“ zurückgenommen).

Über das Menü „Datei“ kann auch zwischen unterschiedlichen Simulationen gewechselt oder neue Simulationen erstellt werden. Hierzu muss der Menüpunkt „Simulation wechseln“ ausgewählt werden. Im sich öffnenden Fenster kann entweder eine bestehende Simulation über ein Drop-Down-Menü ausgewählt oder durch Eingabe eines Namens im entsprechenden Textfeld eine neue Simulation angelegt werden. Zusätzlich kann über den Menüpunkt „Simulation kopieren“ eine bestehende Simulation in eine neue Simulation unter neuem Namen kopiert werden.

Über das Menü „Bearbeiten“ können einige grundlegende Eigenschaften der derzeit aktiven Simulation eingestellt werden. So erlaubt der Menüpunkt „Simulationstitel“ das Einstellen des auf der grafischen Oberfläche sichtbaren Titels der Simulation (blauer Balken ganz oben). Unter „Seiteneinstellungen“ kann die Größe des grauen Hintergrunds (in Punkten) festgelegt werden, auf dem in der grafischen Oberfläche Elemente verteilt werden können.

Zusätzlich kann über das Menü „Bearbeiten“ für die derzeit aktive Simulation zwischen dem Bearbeitungsmodus und dem Simulationsmodus umgeschaltet werden. Nach Auswahl des Menüpunktes „Lock Simulation“ befindet man sich in Simulationsmodus und es taucht am unteren Ende die Simulationssteuerung auf (Abb. A 12). Über „Unlock Simulation“ gelangt man entsprechend wieder in den Bearbeitungsmodus.

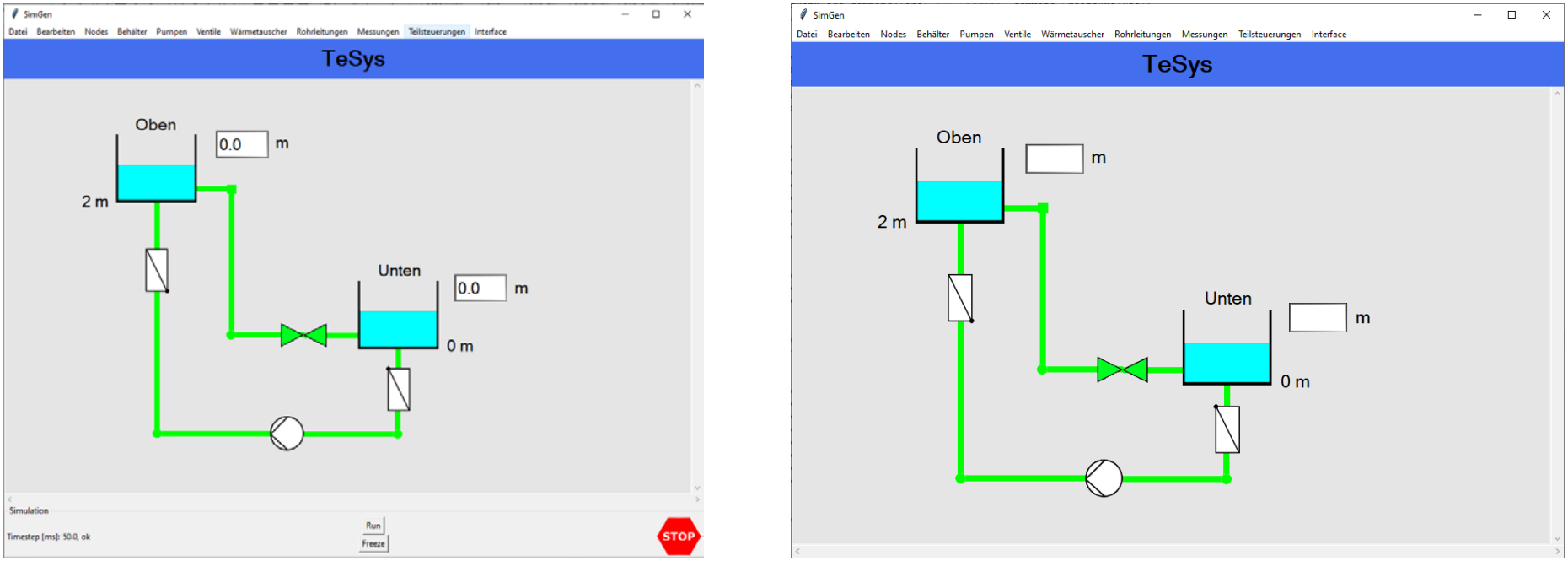


Abb. A 12 Simulationsmodus (links) und Bearbeitungsmodus (rechts) sind leicht durch die nur im Simulationsmodus vorhandene Simulationssteuerung am unteren Ende unterscheidbar.

Die Simulationssteuerung erlaubt das Starten (Button „Run“) und Stoppen (Button „Freeze“) einer Simulation, der jeweilige Zustand wird über das Symbol ganz rechts unten visualisiert (rotes hinterlegtes „Stop“ bzw. grün hinterlegtes „Run“). Zusätzlich wird in der Simulationssteuerung angezeigt, ob die in SimGen festgelegte Zykluszeit von 50 ms für Berechnungen während der Simulation eingehalten wird („Timestep [ms]: 50.0 ok“).

Im Bearbeitungsmodus können Elemente über die obere Menüleiste zur Simulation hinzugefügt oder verwaltet werden. Vorhandene Elemente können im Bearbeitungsmodus über einen Rechtsklick mit der Maus gelöscht werden. Im Simulationsmodus lassen sich Elemente wie in einem Leittechniksystem bedienen (Abb. A 13).

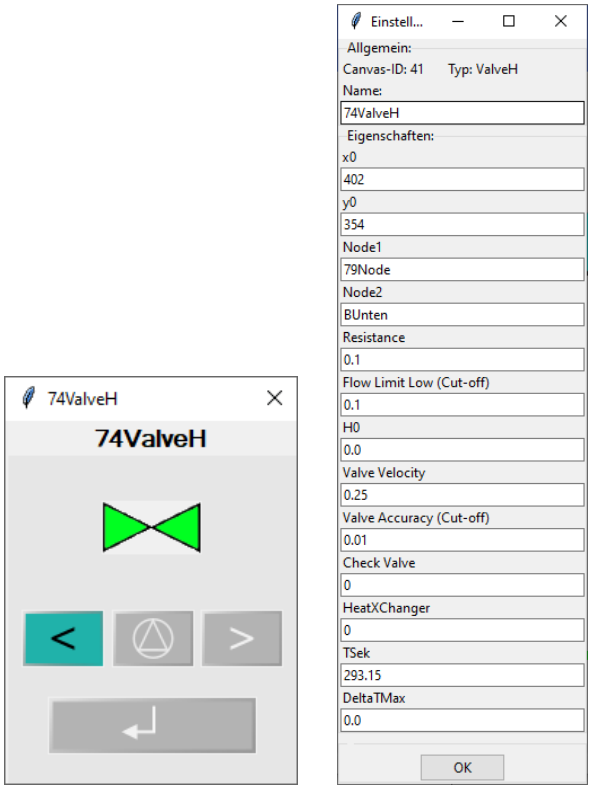


Abb. A 13 Simulationssteuerung – Hinzufügen/Entfernen im Bearbeitungsmodus

Durch einen Doppelklick auf ein Element (in diesem Beispiel das Ventil in Abb. A 12) erreicht man im Simulationsmodus ein Bedienfeld, welches in Anlehnung an typische Leittechniksysteme erstellt wurde (links), im Bearbeitungsmodus wird ein Eigenschaftsfenster für die Eingabe von Simulationsparametern geöffnet (rechts).

Nodes in SimGen

Nodes sind grundlegende Elemente von SimGen, für die zu jedem Zeitpunkt in Simulationen dynamisch Zustände berechnet werden. Über den jeweiligen Füllstand und die definierte Grundfläche eines jeden Nodes ist jederzeit die in ihm gespeicherte Wassermenge (Volumen) und der dort herrschende (statische) Druck definiert. Darüber hinaus wird für jeden Node auch noch die in ihm herrschende Wassertemperatur gespeichert und es besteht die Möglichkeit Heiz- oder Kühlleistungen festzulegen. Die entsprechenden (Start-)Werte für jeden Node können über einen Doppelklick auf das entsprechende Element eingegeben werden (Abb. A 14).

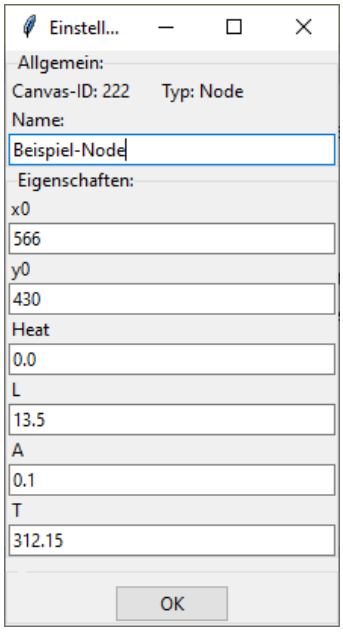


Abb. A 14 Einstellung von Startwerten für Nodes im Eigenschaftsfenster

Die für jeden Node festzulegenden Werte sind im Einzelnen:

Name	Eindeutige Bezeichnung des Nodes
x0, y0	Koordinaten für die grafische Darstellung (alternativ können Objekte in SimGen auch per Drag&Drop mit der Maus positioniert werden)
Heat	Heizleistung in MW
L	Füllstand in m
A	Grundfläche in m²
T	Temperatur in K

Nodes können in SimGen einfache Nodes, Behälter, Poolreaktoren, BE-Lagerbecken oder sogenannte unendliche Nodes sein.

Wichtige Eigenschaften dieser unterschiedlichen Ausprägungen sind nachfolgend aufgelistet:

- Einfache Nodes
  - werden grafisch als grüner Punkt dargestellt,
  - ignorieren eingestellte (interne) Heiz- und Kühlleistungen,
  - die Temperatur wird während der Simulation zu jedem Zeitpunkt durch eine Mischungsrechnung des gespeicherten, abfließenden sowie zufließenden Wassers bestimmt.
- Behälter
  - werden grafisch als Behälter dargestellt, unterscheiden sich ansonsten aber nicht von einfachen Nodes.
- Poolreaktoren
  - werden grafisch als Poolreaktor dargestellt,
  - während der Simulation kann die thermische Heizleistung zwischen dem eingestellten Startwert und 0 MW per Schieberegler dynamisch variiert werden (Einstellung der Reaktorleistung),
  - die Temperatur wird während der Simulation zu jedem Zeitpunkt durch eine Mischungsrechnung des gespeicherten, abfließenden sowie zufließenden Wassers bestimmt; zusätzlich wird die Aufwärmung durch die eingestellte Heizleistung (Reaktorleistung) berücksichtigt.
- BE-Lagerbecken
  - werden grafisch als BE-Lagerbecken dargestellt,
  - die eingestellte Heizleistung stellt die Nachzerfallsleistung dar,
  - die Temperatur wird während der Simulation zu jedem Zeitpunkt durch eine Mischungsrechnung des gespeicherten, abfließenden sowie zufließenden Wassers bestimmt; zusätzlich wird die Aufwärmung durch die eingestellte Heizleistung (Nachzerfallsleistung) berücksichtigt.
- Unendliche Nodes
  - werden grafisch als grüner Punkt dargestellt,
  - haben einen festen Füllstand (und damit auch festen statischen Druck),
  - können beliebig viel Wasser aufnehmen bzw. abgegeben,
  - werden typischerweise für Grenzen zu Systemen verwendet, die nicht explizit simuliert werden.

Connections in SimGen

Connections in SimGen stellen jeweils eine Verbindung zwischen zwei Nodes dar, über die diese Wasser austauschen können. Die für jede dieser Connections gespeicherten Werte sind beispielhaft in Abb. A 15 dargestellt.

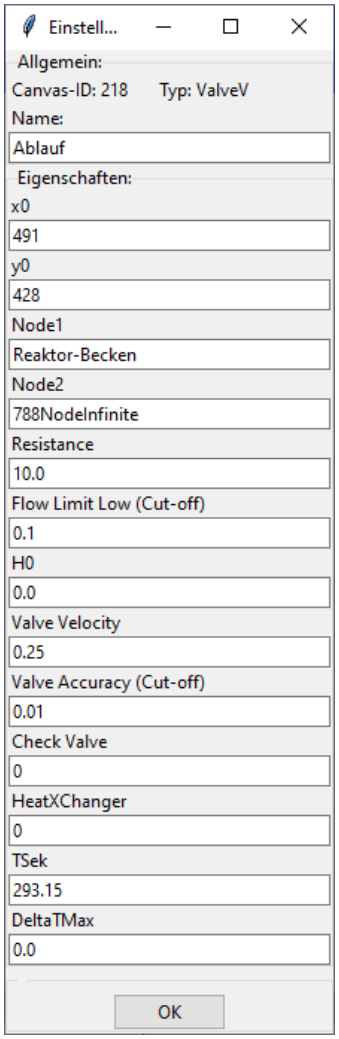


Abb. A 15 Beispiel für Einstellwerte von Connections in SimGen (hier ein vertikales Ventil).

Die für jede Connection festzulegenden Werte sind im Einzelnen:

Name	Eindeutige Bezeichnung der Verbindung
x0, y0	Koordinaten für die grafische Darstellung (alternativ können Objekte in SimGen auch per Drag&Drop mit der Maus positioniert werden)
Node1	Name des ersten angebundenen Nodes
Node2	Name des zweiten angebundenen Nodes

Resistance

„Rohrleitungswiderstand“:

Bei der Berechnung der Durchflussmengen wird hier vereinfachend angenommen, dass Wasser vollständig inkompressibel ist und sich auch dessen Dichte mit der Temperatur nicht ändert. Für die Durchflussmenge Q (in m^3/s) einer Verbindung gilt dann die folgende Beziehung (wobei *p* für Druck und *L* für Level steht):

$$Q \sim \sqrt{\Delta p} \sim \sqrt{\Delta L}$$

Legt man als Proportionalitätskonstante den Wert K fest, ergibt sich:

$$Q = K \cdot \sqrt{\Delta L} = \sqrt{\frac{\Delta L}{1/K^2}} = \sqrt{\frac{\Delta L}{Resistance}}$$

Die auf diese Weise definierte „*Resistance*“ hat formal also die Dimension s/m^5. Für die mit SimGen generierten Simulationen wurden die Werte für *Resistance* jeweils empirisch derart ausgewählt, dass die Durchflussmengen gerade den gewünschten Mengen entsprachen.

Flow Limit Low

Unterhalb des hier eingestellten Cut-Off-Wertes wird der Durchfluss durch die jeweilige Connection auf null gesetzt. Es handelt sich um ein numerisches Hilfsmittel, damit Durchflüsse (z. B. in Ventilen) stabil bis auf 0 m^3/s abgesenkt werden können.

H0

Nullförderhöhe in m (für Pumpen). Formal wird der hiermit eingestellte Wert auf die Druckdifferenz zwischen den beiden verbundenen Nodes aufaddiert, wobei die Druckdifferenz hier (in m) als Differenz der Füllstände berechnet wird.

Valve Velocity

Legt fest, wie schnell Ventile verfahren werden können (relative Größe). Beispielsweise kann ein Ventil mit einem Wert von 0,25 in 4 Sekunden vollständig geschlossen werden.

Valve Accuracy

Numerische Hilfsgröße (Cut-Off-Wert), damit Ventile in der Simulation vollständig geschlossen werden können.

Check Valve

Wird hier anstatt einer „0“ eine „1“ angegeben, verhält sich die Verbindung als Rückschlagklappe. In diesem Fall können Durchflüsse nur noch von Node1 nach Node2 auftreten, umgekehrte Durchflüsse werden unterdrückt (auf „0“ gesetzt).

HeatXChanger	Wird hier anstatt einer „0“ eine „1“ angegeben, verhält sich die Connection als Wärmetauscher. In diesem Fall wird das durchfließende Wasser in Richtung der Temperatur der Sekundärseite abgekühlt, jedoch maximal um den unter DeltaTMax angegebenen Wert.
TSek	Temperatur auf der Sekundärseite, falls die Connection ein Wärmetauscher ist. Ist der Wert unter „HeatXChanger“ auf null gesetzt, wird diese Eingabe ignoriert.
DeltaTMax	Maximale Aufwärm- oder Abkühlspanne (in K), falls die Connection ein Wärmetauscher ist. Indirekt legt dieser Wert einen genäherten Wirkungsgrad in Abhängigkeit der aktuellen Temperaturdifferenz zwischen Primär- und Sekundärseite für den entsprechenden Wärmetauscher fest. Ist der Wert unter „HeatXChanger“ auf null gesetzt, wird diese Eingabe ignoriert.

Connections können in SimGen Rohrleitungen, Ventile, Rückschlagklappen, Pumpen oder Wärmetauscher sein. Wichtige Eigenschaften dieser unterschiedlichen Ausprägungen sind nachfolgend aufgelistet:

- Rohrleitung
  - wird grafisch als grünes Viereck dargestellt.
- Ventil
  - Ventile werden mit einem entsprechenden Symbol dargestellt. Diese werden je nach Öffnungszustand des Ventils weiß (geschlossen), grün (vollständig geöffnet) oder halb-grün-halb-weiß (teilweise geöffnet) dargestellt. Darüber hinaus blinken Ventile beim Öffnen oder Schließen (d. h. sie werden abwechselnd halb-grün-halb-weiß und dunkelgrau hinterlegt im angestrebten Zustand (weiß – geschlossen, grün – offen) dargestellt).
- Rückschlagklappe
  - Rückschlagklappen werden mit einem entsprechenden Symbol dauerhaft in weißer Farbe dargestellt.
- Pumpe
  - Pumpen werden entweder in grün (eingeschaltet) oder weiß (abgeschaltet) mit einem entsprechenden Symbol dargestellt.
- Wärmetauscher
  - Wärmetauscher werden dauerhaft weiß mit einem entsprechenden Symbol dargestellt.

Connections werden also durch eigene (teilweise dynamische) grafische Symbole dargestellt. Diese werden nach der Verbindung mit Nodes (über ihre jeweiligen Eigenschaftsfenster) zusätzlich automatisch grafisch über grüne Verbindungslinien mit den entsprechenden Nodes verbunden. Diese Verbindungslinien werden beim Verschieben von Elementen automatisch aktualisiert.

Anbindung von SimGen an AnTeS

Für die Anbindung einer Simulation an die Leittechniksysteme von AnTeS können in SimGen Messungen für jeden beliebigen Node über die obere Menüleiste erstellt werden. Diese sind nicht nur mit der Leittechnik koppelbar (siehe weiter unten), sondern werden auch auf der grafischen Oberfläche von SimGen während der Simulation dynamisch mit aktuellen Messwerten dargestellt. Messungen (bzw. deren Messwertdarstellung) können in SimGen über ihre jeweiligen Eigenschaftsfenster (Abb. A 16) mit einem Node verbunden werden (durch Eingabe des Namens des entsprechenden Nodes unter „Node1“ in diesem). Darüber hinaus erlaubt das Eigenschaftsfenster die Festlegung des Namens der Messung sowie die punktgenaue Positionierung der Messwertdarstellung auf der grafischen Oberfläche von SimGen (welche auch mit der Maus per Drag&Drop erfolgen kann).

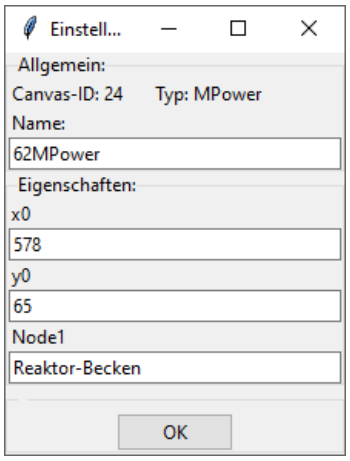


Abb. A 16 Beispiel für ein Eigenschaftsfenster einer Messung (hier: Leistungsmessung am Node „Reaktor-Becken“) in SimGen.

Derzeit stehen in SimGen die folgenden Typen von Messungen zur Verfügung:

- Druck
  - Die Ausgabe an die Leittechnik und Darstellung auf der grafischen Oberfläche erfolgt in bar (absolut).
- Füllstand
  - Die Ausgabe an die Leittechnik und Darstellung auf der grafischen Oberfläche erfolgt in m.
- Durchfluss
  - Die Ausgabe an die Leittechnik und Darstellung auf der grafischen Oberfläche erfolgt in kg/s.
- Temperatur
  - Die Ausgabe an die Leittechnik und Darstellung auf der grafischen Oberfläche erfolgt in °C.
- Leistung
  - Die Ausgabe an die Leittechnik und Darstellung auf der grafischen Oberfläche erfolgt in %.
  - 100 % sind dabei durch die Eingabe der Heizleistung für den jeweiligen Node im entsprechenden Eigenschaftsfenster definiert.

Zusätzlich zu den Messungen ist im Menü „Messungen“ noch der Menüpunkt „Text“ zu finden. Über diesen können reine Textbausteine (die über entsprechende Eigenschaftsfenster konfigurierbar sind) auf der grafischen Oberfläche von SimGen erstellt werden (beispielsweise für die Beschriftung einer Messung).

Daneben gibt es in der oberen Menüleiste von SimGen noch das Menü „Teilsteuerungen“. Hierüber können Teilsteuerungen erstellt werden, die ebenfalls auf der grafischen Oberfläche von SimGen über ein entsprechendes Symbol dargestellt werden. Bei Teilsteuerungen handelt es sich um vergleichsweise einfache Bausteine, die lediglich zwischen zwei Zuständen umgeschaltet werden können („Ein“ – Darstellung mit grünem Symbol bzw. „Aus“ – Darstellung mit weißem Symbol).

Die Anbindung einer Simulation an ein leittechnisches System (von AnTeS) über die AnTeS-Interfaces erfolgt über das Menü „Interface“ in der oberen Menüleiste von SimGen (Abb. A 17).

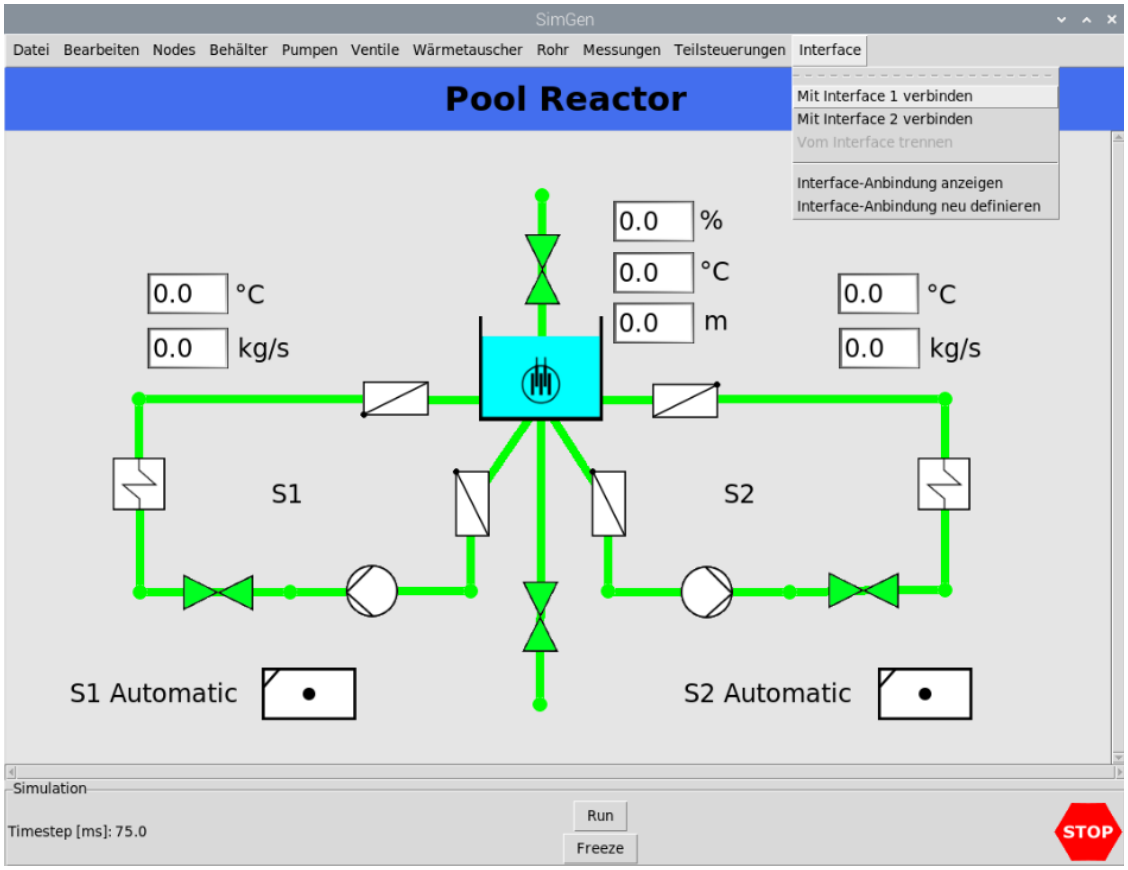


Abb. A 17 Anbindung der Interfaces über das entsprechende Menü von SimGen.

Vor dem Verbinden mit den Interfaces muss zuvor festgelegt werden, welche Messungen und Rückmeldesignale sowie welche Leittechnikausgangssignale welchen Steuersignalen entsprechen sollen. Hierzu dient der Menüpunkt „Interface-Anbindung neu definieren“ (der Menüpunkt „Interface-Anbindung“ zeigt die derzeit gültige Definition lediglich an). Nach dessen Auswahl erscheint das Fenster in Abb. A 18. Dort können über Drop-Down-Menüs Signale der jeweiligen Simulation für die einzelnen Ein- und Ausgänge ausgewählt werden sowie für die analogen Ausgangssignale auch Kalibrierungswerte eingestellt werden.

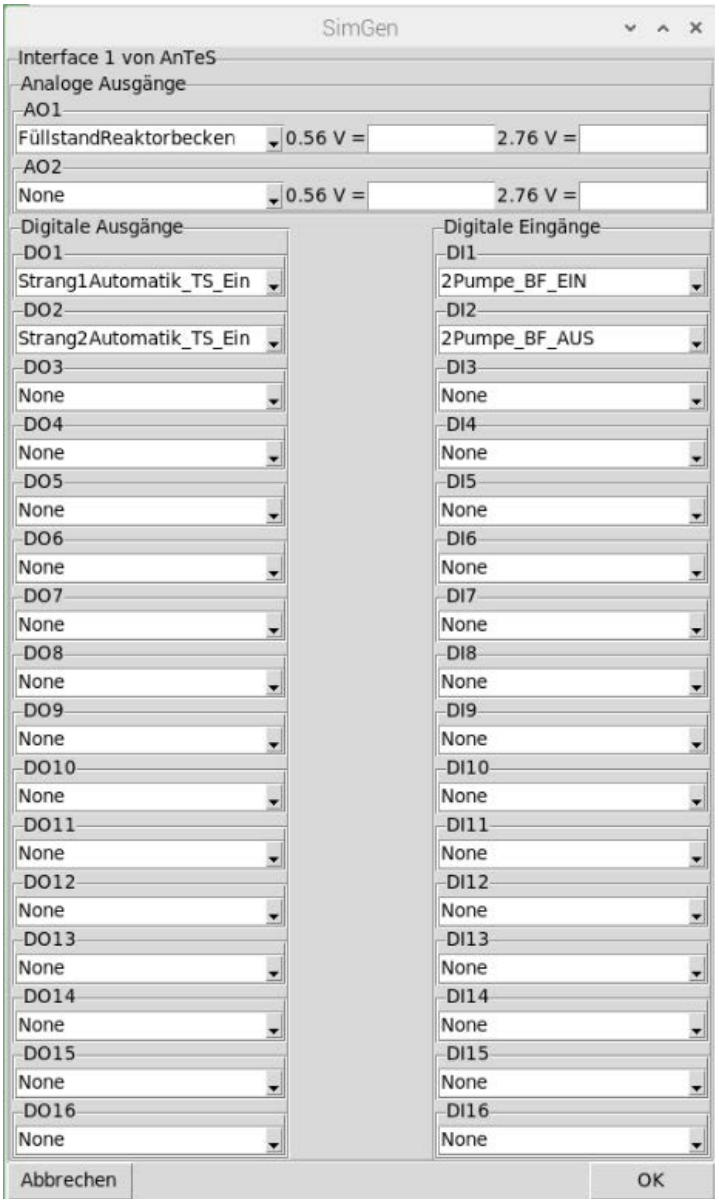
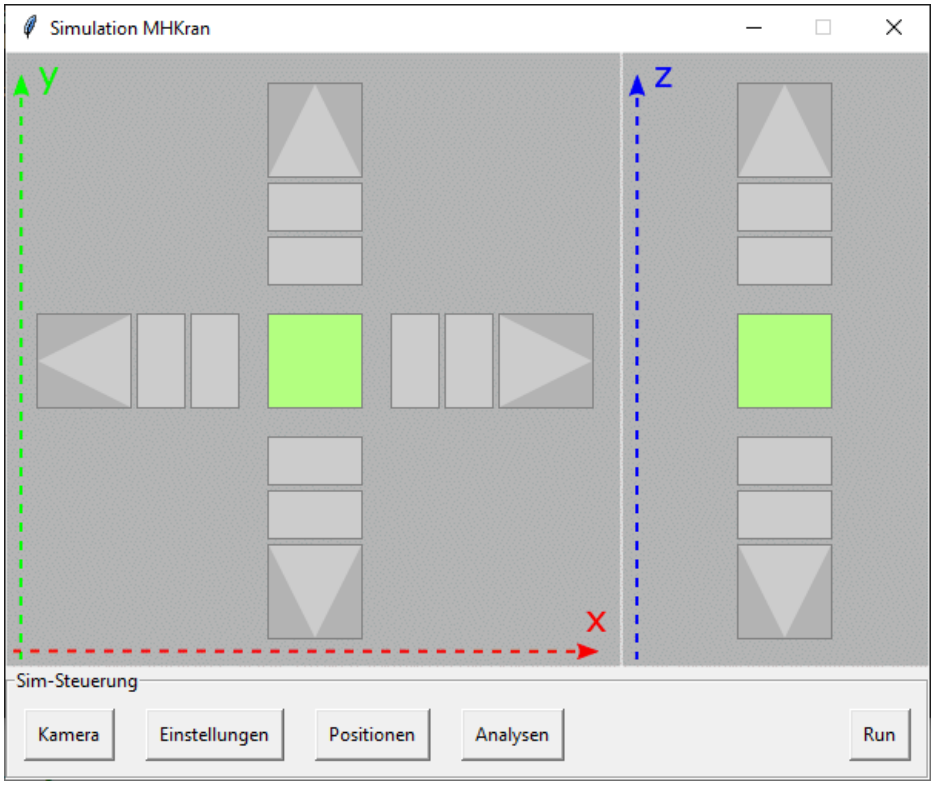


Abb. A 18 Einstellungen für die Ein- und Ausgänge der Interfaces in SimGen (Anmerkung: die ansteuerbaren Ausgänge sind mit Eingängen des Leittechniksystems verbunden und umgekehrt).

A.5 Detaillierte Beschreibung von KranSim

Nach dem Start der Software öffnen sich zwei Fenster, eines für die Steuerung der Simulation (Abb. A 19, links) und eines für die dynamische 3D-Visualisierung (Abb. A 19, rechts). Über den Button „Run“ des Steuerungsfensters wird die Simulation gestartet, im laufenden Zustand ist dann der Rahmen des Fensters ausgeblendet und die Beschriftung des Buttons ändert sich zu „Freeze“. Ein erneutes Betätigen des Buttons („Freeze“) stoppt die Simulation (Abb. A 20).



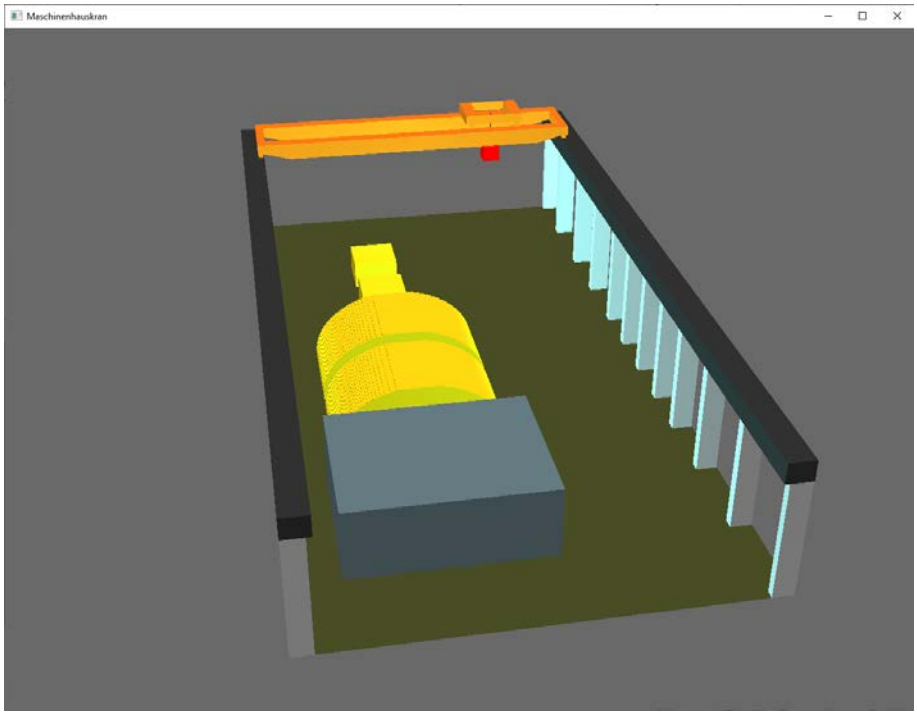


Abb. A 19 Screenshots von der Software KranSim, die von der GRS zur Simulation eines Maschinenhauskrans erstellt wurden.

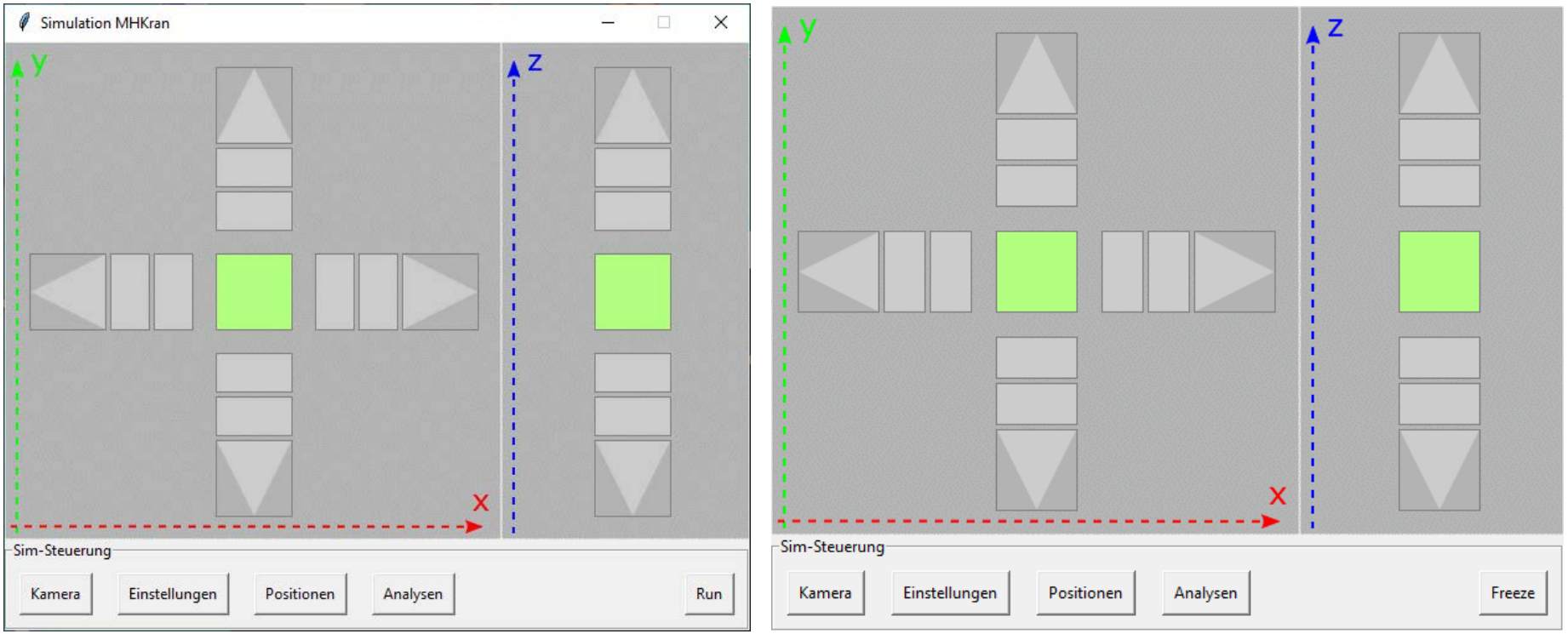


Abb. A 20 Das Steuerungsfenster von KranSim bei gestarteter (rechts) und angehaltener Simulation (links)

Über den Button „Kameras“ des Steuerungsfensters gelangt man zu einer Auswahl unterschiedlicher Ansichtsmöglichkeiten des Visualisierungsfensters, diese haben die folgenden Eigenschaften (Abb. A 21):

- „Freie Kamera“
  - Die (virtuelle) Kamera des Visualisierungsfensters kann frei bewegt werden, die Bedienung erfolgt über die Tastatur:
    - „w“ – Kamera bewegt sich nach vorne
    - „a“ – Kamera bewegt sich nach links
    - „s“ – Kamera bewegt sich nach hinten
    - „d“ – Kamera bewegt sich nach rechts
    - „r“ – Kamera bewegt sich nach oben
    - „f“ – Kamera bewegt sich nach unten
    - „t“ – Kamera schwenkt nach oben
    - „g“ – Kamera schwenkt nach unten
- „Mensch“
  - Die (virtuelle) Kamera befindet sich fest in Kopfhöhe über dem Maschinenausboden.
  - Ansonsten verhält sich die (virtuelle) Kamera genauso wie im Modus „Freie Kamera“, allerdings kann diese nicht über „r“ und „f“ nach oben oder unten bewegt werden.
  - Man bewegt sich also virtuell wie ein Mensch durch das Maschinenhaus.
- „Führerhaus“
  - Die (virtuelle) Kamera befindet sich in fester Höhe unter der Laufkatze des Maschinenhauskrans, als ob man sich in einem Führerhaus befände, und bewegt sich bei Bewegungen des Krans mit diesem mit.
  - Die Blickrichtung ist auf die angehängte Last ausgerichtet, die Kamera kann lediglich nach oben („r“) oder unten („f“) geschwenkt werden.
- Anmerkungen:
  - Programmiertechnisch erfolgt die Steuerung über das Steuerungsfenster (Abb. A 19, links), d. h. dieses Fenster muss in Windows das aktive Fenster sein, damit die Tastatureingaben funktionieren
  - Die Simulation muss laufen (also über den Button „Run“ gestartet worden sein), damit die Tastatureingaben funktionieren.

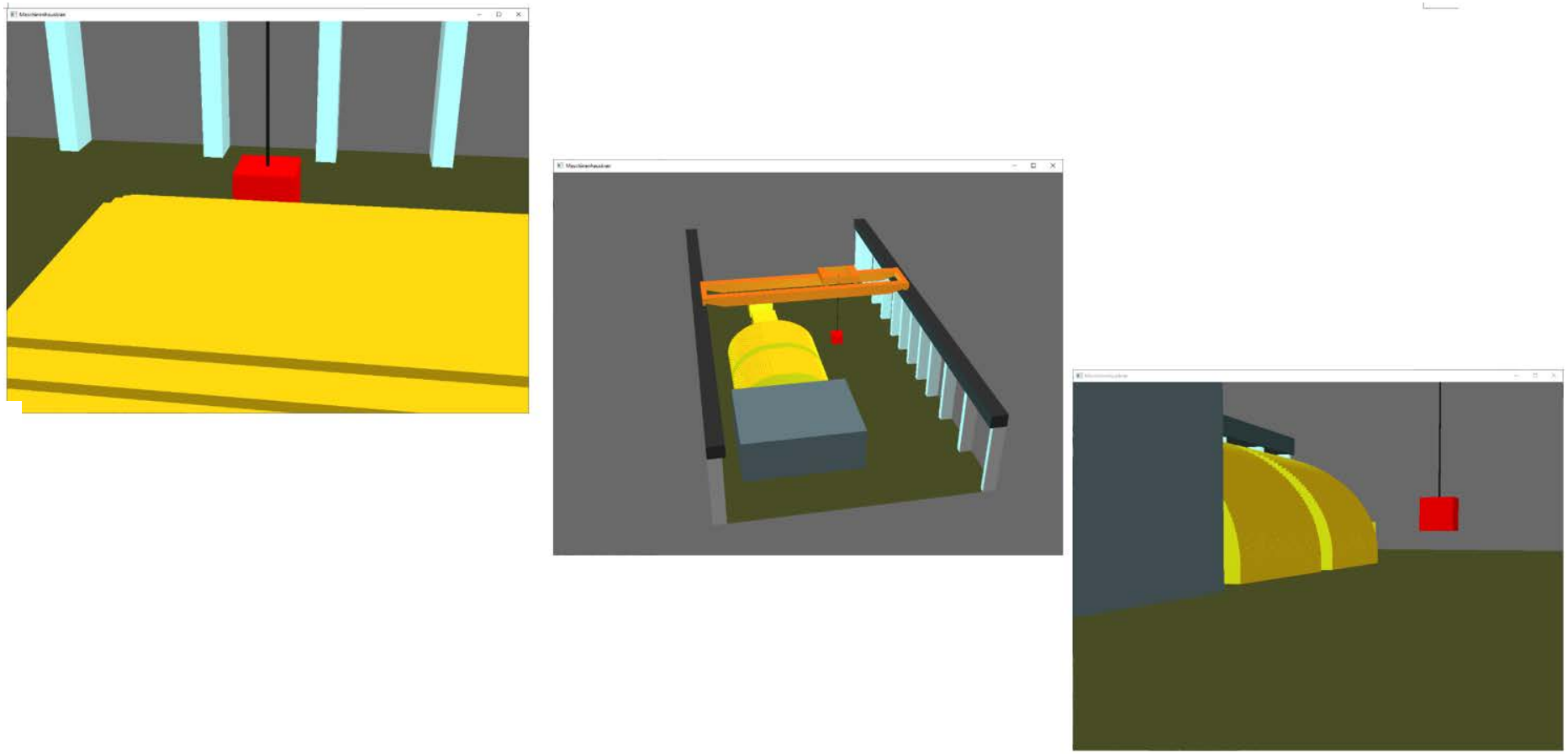


Abb. A 21 Verschiedene Ansichtsmöglichkeiten in KranSim (von links nach rechts: „Freie Kamera“, „Mensch“, „Führerhaus“)

Die Steuerung des Krans im Handbetrieb kann über die Bedienfelder des Steuerungsfensters in Echtzeit erfolgen. Hierzu stehen in den drei Raumrichtungen (x, y, z) jeweils drei unterschiedliche Fahrgeschwindigkeiten zur Verfügung, die unter „Einstellungen“ im Steuerungsfenster verändert werden können (dort ist auch eine Veränderung der Kamerageschwindigkeiten möglich). Angesteuert wird das Verfahren des Krans durch das Anklicken der entsprechenden Symbole für die unterschiedlichen Bewegungsrichtungen und Geschwindigkeiten (Abb. A 22). Dabei können Bewegungen in x-, y- und z-Richtung (durch Mehrfachauswahl) gleichzeitig durchgeführt werden. Das Anklicken eines der beiden zentralen Quadrate stoppt sämtliche Kranbewegungen.

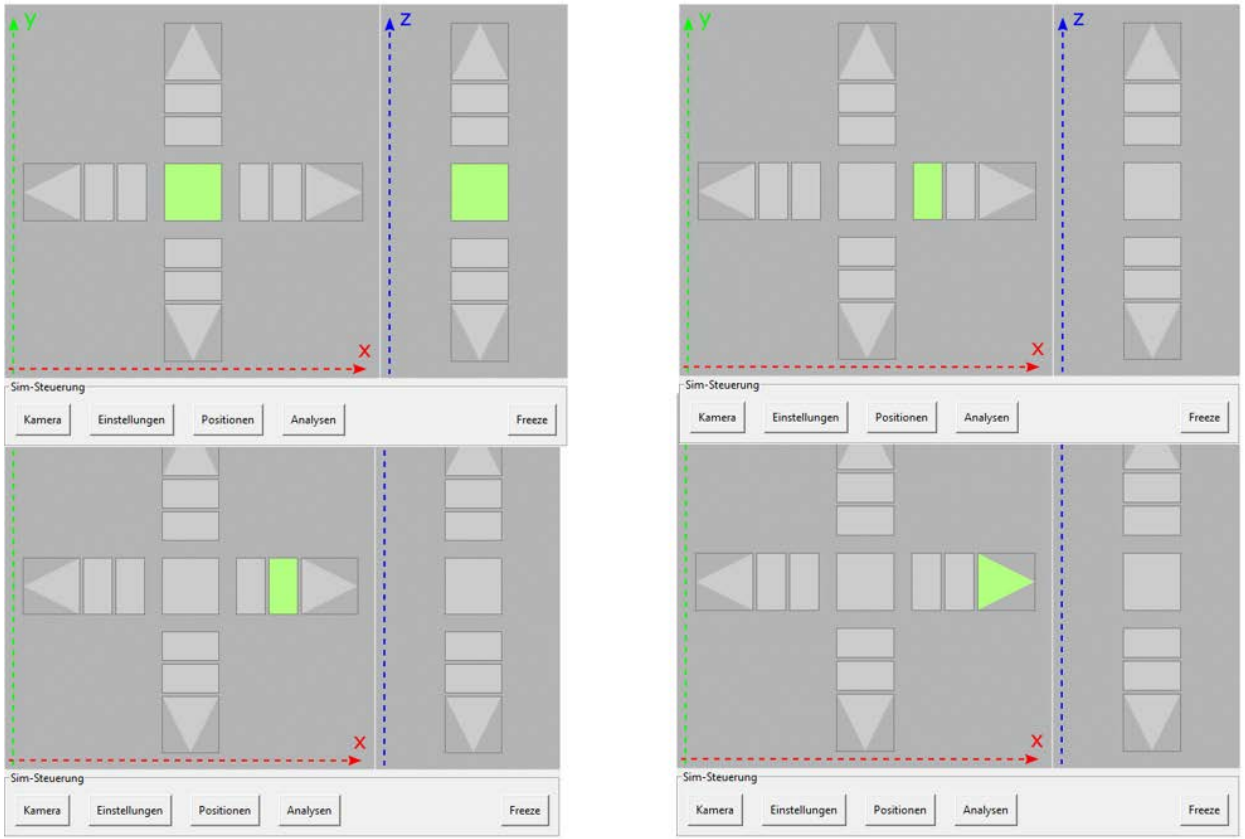


Abb. A 22 Kranbewegungen mit unterschiedlichen Geschwindigkeiten

Das Steuerungsfenster von KranSim erlaubt das Bewegen des Krans mit unterschiedlichen Geschwindigkeiten (beispielhaft von links nach rechts: Kran steht, Kran bewegt sich mit langsam in x-Richtung, Kran bewegt sich mit mittlerer Geschwindigkeit in x-Richtung, Kran bewegt sich mit hoher Geschwindigkeit in x-Richtung).

Über den Button „Positionen“ des Steuerungsfensters kann ein Fenster aufgerufen werden, auf welchem Positionen des Krans (bzw. der angehängten Last) gespeichert bzw. wieder aufgerufen werden können (Abb. A 23). Zusätzlich kann auf diesem Fenster auch der Kran in die Standardposition (wie beim Softwarestart) versetzt und ein gleichzeitiger Neustart der Simulation über einen entsprechenden Button erfolgen.

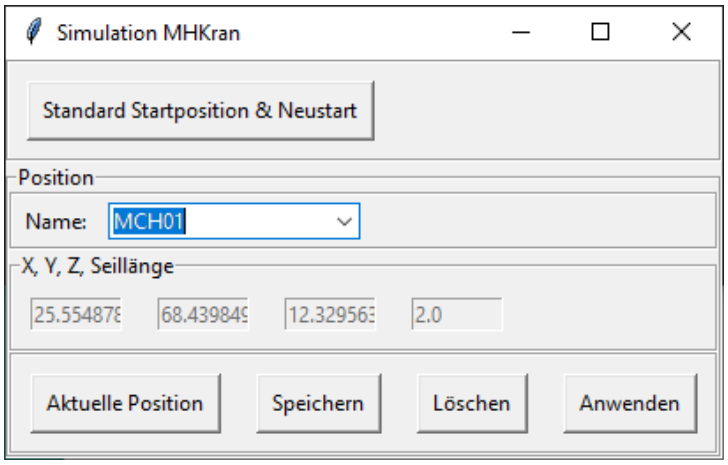


Abb. A 23 Über den Button „Positionen“ des Steuerungsfensters kann ein Fenster aufgerufen werden, dass das Speichern und Wiederaufrufen von Kranpositionen erlaubt.

KranSim erlaubt einige automatische Analysen, diese sind über den Button „Analysen“ des Steuerungsfensters erreichbar. Betätigt man diesen, so öffnet sich das in Abb. A 24 dargestellte Fenster, auf welchem Einstellungen für die Analysen eingegeben und diese auch gestartet werden können.

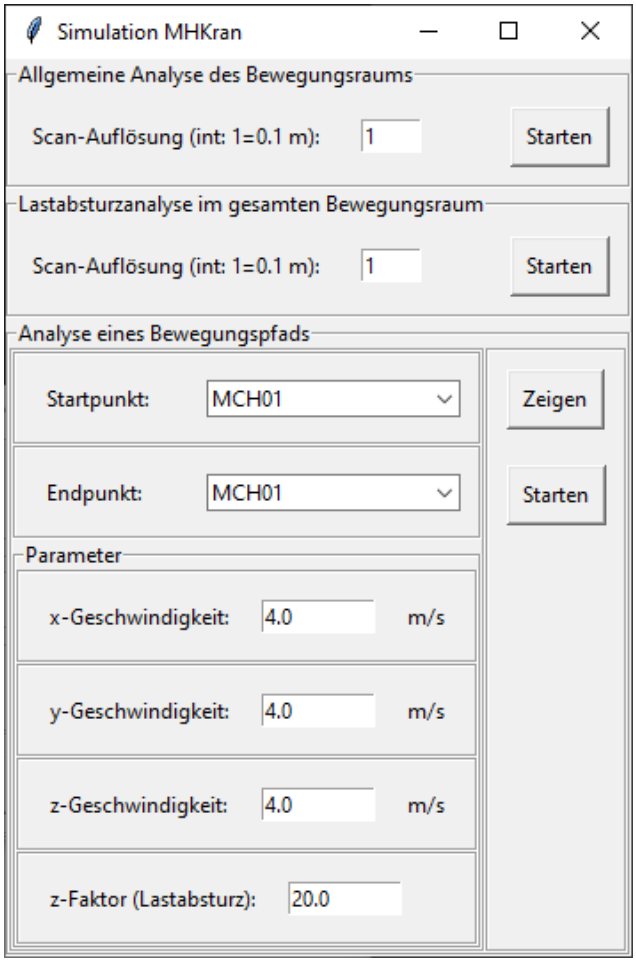


Abb. A 24 Eingabefenster für automatische Analysen mit KranSim.

Die Analysemöglichkeiten in KranSim umfassen allgemeine und Lastabsturzanalysen des gesamten Bewegungsraums sowie Analysen entlang eines frei definierbaren Bewegungspfads des simulierten Krans. Die bei automatischen Analysen durchgeführten Kranbewegungen erfolgen wesentlich schneller als in Echtzeit. Eine genauere Beschreibung dieser Analysemöglichkeiten erfolgt im Abschnitt 4.3.2 im Rahmen der Beschreibung der mit AnTeS durchgeführten Tests.

Der Bewegungsraum des Maschinenhauskrans (also das simulierte Maschinenhaus mit allen darin befindlichen Hindernissen) kann für KranSim flexibel variiert werden. Hierfür steht eine separate von der GRS (ebenfalls in der Programmiersprache Python) entwickelte Software zur Verfügung (Abb. A 25). Die entsprechenden Einstellungen werden im Installationsverzeichnis von KranSim in einer eigenen SQLite-Datenbank verwaltet.

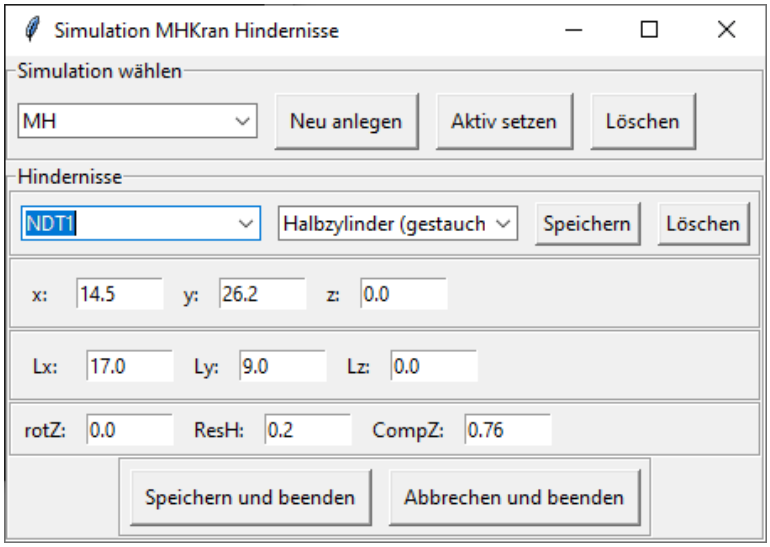


Abb. A 25 Spezielle Software für Hindernisse im Bewegungsraum

Der Bewegungsraum (insbesondere die Hindernisse im Maschinenhaus) des in KranSim simulierten Maschinenhauskrans kann über eine eigene Software konfiguriert werden.

Derzeit können hierfür frei positionierbare Quader und stauchbare Halbzylinder für unterschiedliche Simulationskonfigurationen verwendet werden. Dieses Programm erlaubt auch das einfache Wechseln zwischen Simulationen oder das Anlegen von neuen Simulationskonfigurationen (im Bereich „Simulation wählen“). Jedes Hindernis einer Simulationskonfiguration wird durch neun Parameter definiert, diese sind im Einzelnen:

x, y, z	Längs-, Quer- und Höhenposition des Hindernisses in m (relativ zum simulierten Maschinenhaus).
Lx, Ly, Lz	Längen des Hindernisses in x-, y- und z-Richtung (bevor dieses evtl. noch rotiert wird – siehe Parameter „rotZ“). Für (stauchbare) Halbzylinder wird der Wert von Lz ignoriert, dessen Länge in z-Richtung ergibt sich aus dem Parameter CompZ.
rotZ	Rotation des Hindernisses um die z-Achse (diese weist nach „oben“).
ResH	Legt für (stauchbare) Halbzylinder die Höhe der Quader in m fest, durch die die Zylinderform angenähert wird.
CompZ	Legt die Stauchung eines Halbzylinders fest. Ein Wert von 1 für CompZ erzeugt einen Zylinder ohne Stauchung („perfekter“ Halbzylinder), Werte < 1 führen zu einer entsprechenden Verkürzung in z-Richtung, Werte > 1 zu in z-Richtung langgezogenen Halbzylindern.

Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH

Schwertnergasse 1
50667 Köln
Telefon +49 221 2068-0
Telefax +49 221 2068-888

Forschungszentrum
Boltzmannstraße 14
85748 Garching b. München
Telefon +49 89 32004-0
Telefax +49 89 32004-300

Kurfürstendamm 200
10719 Berlin
Telefon +49 30 88589-0
Telefax +49 30 88589-111

Theodor-Heuss-Straße 4
38122 Braunschweig
Telefon +49 531 8012-0
Telefax +49 531 8012-200

www.grs.de

ISBN 978-3-949088-37-7