Humboldt-Universität zu Berlin
Institut für Informatik

Formale Spezifikation in Reverse-Engineering-Prozessen für technische Systeme

Diplomarbeit

eingereicht von Jörg Lange

Berlin, den 21. Dezember 2006

Gutachter: Prof. Dr. Bothe, Prof. Dr. Reisig

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Berlin, den 21. Dezember 2006

Jörg Lange

Einverständniserklärung

Ich erkläre hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Institutes für Informatik der Humboldt-Universität zu Berlin ausgestellt werden darf.

Berlin, den 21. Dezember 2006

Jörg Lange

Überblick

Im Rahmen des Projektes „Software-Sanierung“ am Institut für Informatik der Humboldt Universität zu Berlin wird ein Softwaresystem des Instituts für Physik der Humboldt Universität zu Berlin einem Reverse-Engineering-Prozess unterzogen. Bislang fanden innerhalb des Projektes formale Beschreibungstechniken bei der Erstellung von Dokumenten für den Reverse-Engineering-Prozess keine Anwendung. In der Diplomarbeit werden die Erfahrungen und Ergebnisse bei der Durchführung der formalen Spezifikation von zwei Teilsystemen des Softwaresystems dokumentiert. Die Spezifikationssprache, mit der die Modellierung erfolgte, ist die Z-Notation. Auf der Grundlage der entwickelten formalen Spezifikationen werden die vorliegenden verbalen Verhaltensspezifikationen im Hinblick auf Probleme bezüglich Mehrdeutigkeit bzw. Ungenauigkeit untersucht.

1. Problemstellung

1.1 Das Projekt Softwaresanierung

Am Lehr- und Forschungsgebiet „Softwaretechnik“ des Institutes für Informatik der Humboldt Universität zu Berlin findet seit 1998, unter der Leitung von Professor Bothe, das Projekt „Software-Sanierung“ statt [1]. Gegenstand ist eine Software des Instituts für Physik der Humboldt Universität zu Berlin.
Die Arbeitsgruppe „Röntgenstrahlung an Schichtsystemen“, geleitet von Professor Köhler, nutzt die Software zur Steuerung von Messplätzen. Untersuchungsobjekte sind kristalline Strukturen von Halbleitern. Abbildung 1 zeigt einen solchen Messplatz.

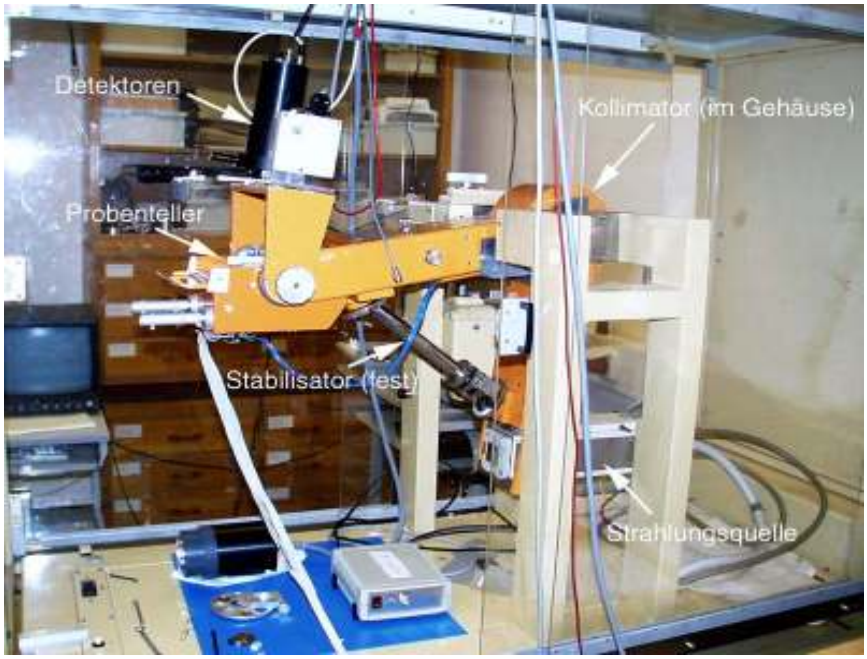
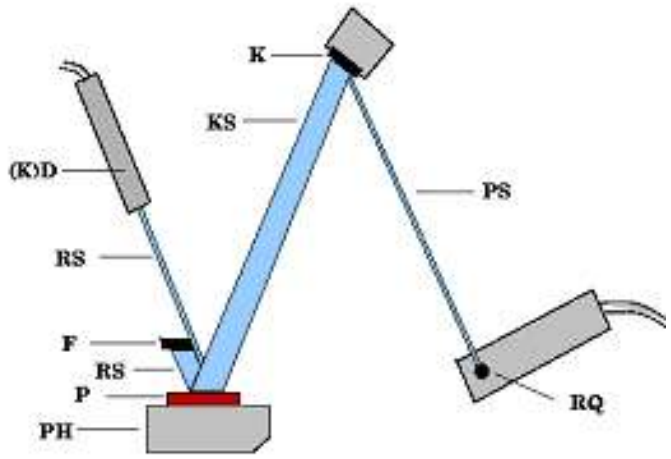


Abb. 1: Messplatz Topographiekamera[1]

An den Messplätzen können unterschiedliche Eigenschaften kristalliner Strukturen untersucht werden. Der Messvorgang findet, vereinfacht dargestellt, wie folgt statt: Ein elektromagnetischer Strahl, im speziellen Fall ein Röntgenstrahl, wird auf ein Hilfskristall gerichtet (Kollimator). Der Kollimator fächert den Strahl auf, der anschließend auf die zu untersuchende Kristallprobe trifft. Aufgrund der inneren Struktur der Probe wird der Strahl gebeugt oder reflektiert. Die gebeugten oder reflektierten Strahlen werden von Detektoren erfasst und von der XCTL-Software (**X**-Ray **C**on**t**ro**l**) gesichert.

[1] Diese und die Abbildungen 2, 3 und 4 sind der Internetseite des XCTL-Projektes entnommen. [XCTL xx]

In Abbildung 2, ist eine schematische Darstellung eines Messplatzes zu sehen. Die Probe, die sich auf einem Probenteller befindet und der gefächerte Strahl müssen für eine Messung in einer bestimmten Position zueinander stehen. Die Probe kann durch elektromechanische Antriebe in eine gewünschte Position gebracht werden. Der Kollimator kann ebenfalls durch einen Antrieb gebeugt werden und somit dem Strahl eine bestimmte Richtung geben.



RQ - Röntgenquelle
PS - Primärstrahl
K - Kollimator
KS - kollimierter Strahl
PH - Probenhalter
P - Probe
RS - reflektierter Strahl
F - Fotoplatte
(K)D - (Kontroll-) Detektor

Abb. 2: Schematische Darstellung der Zwei-Kristall-Topographiekamera

Für die Steuerung der Antriebe, die Aufnahme und Umwandlung der Messdaten wird das XCTL-Programm genutzt.
Die Software wurde von Heiko Damerow und Heiko Brandhorst entwickelt (bis 1998). Weiterentwickelt wurde das Programm im Rahmen des Projektes Software Sanierung durch viele am Projekt beteiligte Sudenten.
Die Software wurde und wird unter anderem einem intensiven Reverse-Engineering-Prozess unterzogen, infolge dessen eine Reihe von Dokumenten entstand, die in den einzelnen Entwicklungsphasen der Software nicht erstellt wurden.

1.2 Reverse-Engineering

Der Begriff des Reverse-Engineering bezieht sich auf den Prozess der Analyse eines bestehenden Systems. Die Komponenten des Systems sowie ihre Beziehungen zueinander sollen identifiziert und beschrieben werden. Genutzt wird das Reverse-Engineering, um eine Kopie des Systems zu erstellen oder das

System zu überarbeiten, um eine mögliche Handhabung zu verbessern[2]. Ein weiteres Einsatzgebiet bietet die Dokumentation von Software; nicht vorhandene oder unvollständige Dokumente können erstellt bzw. vervollständigt werden. Außerdem kann eine Anpassung an aktuell geforderte Dokumentstandards vorgenommen werden.
Im Fall des XCTL-Systems standen den Mitgliedern des Projektes die Quellen der Software zur Verfügung, die jedoch ursprünglich kaum kommentiert waren. Dokumente, die im Rahmen eines guten Softwareentwicklungsprozesses entstehen, wie zum Beispiel ein Pflichtenheft, existierten jedoch nicht [vgl. 1].

1.3 Formale Methoden

Bislang wurden formale Spezifikationstechniken innerhalb der Projektarbeit nicht genutzt, um das Verhalten des XCTL-Systems zu beschreiben. Die existierenden Dokumente wurden in verbaler Form erstellt und werden regelmäßig einem Reviewprozess unterzogen, um ihre Qualität zu verbessern. Die menschliche Sprache ist jedoch unpräzise und mehrdeutig. Dieser Nachteil kommt auch hier zum Tragen. Formale Methoden als alternative Beschreibungstechniken sind demgegenüber präzise und eindeutig. Bei der Entwicklung von Software bietet der Einsatz formaler Methoden die Möglichkeit diese zu verbessern, was auch nötig ist, sieht man sich bekannte Beispiele für das Versagen an: Die Ariane 5 Rakete explodierte, weil die Umwandlung einer 64 Bit Fließkommazahl in eine 16 Bit Integer scheiterte[3].
Der Begriff „Formale Methoden“ wird auf der Internetseite: NASA Langley Formal Methods Site[4] beschrieben: „’Formale Methoden’ beziehen sich auf streng mathematische Techniken und Werkzeuge für Spezifikation, Design und Verifikation von Software und Hardwaresystemen. …“[5].
Einsatz finden formale Methoden häufig im Bereich sicherheitskritischer Softwaresysteme, da bewiesen werden kann, dass die Spezifikation widerspruchsfrei ist oder dass bestimmte Systemeigenschaften erfüllt sind. Der Aufwand und die Kosten bei der Softwareentwicklung erhöhen sich durch den Einsatz von formalen Methoden, insbesondere durch den Zusatzaufwand bei der Erstellung formaler Spezifikationen sowie bei eventuellen Beweisen. Daher werden formale Methoden für die Entwicklung von nicht sicherheitsrelevanten Systemen selten genutzt. Doch auch ohne den Beweis von Systemeigenschaften

[2] vgl. Free Online Dictionary of Computing, http://foldoc.org/
[3] http://www.ima.umn.edu/~arnold/disasters/ariane.html
[4] http://shemesh.larc.nasa.gov/fm/index.html
[5] Übersetzt aus dem Englischen: “"Formal Methods" refers to mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems….”

könnten formale Spezifikationen Vorteile gegenüber der bisherigen Praxis bringen (vgl. [2]).
Eine Vielzahl formaler Methoden steht zur Verfügung. An dieser Stelle möchte ich auf die „Formal Methods Homepage“ aufmerksam machen. Dort findet sich eine beachtliche Auswahl an formalen Methoden. Die Internetseite wird von Professor Jonathan Bowen, Gründer und Leiter des „Centre for Applied Formal Methods“[6] und derzeit Gastdozent der South Bank Universität London, betreut.

1.4 Die Z-Notation

Die in dieser Arbeit genutzte formale Spezifikationsmethode ist die Z-Notation. Benannt ist diese Methode nach der Zermelo-Fraenkel-Mengenlehre. Z wurde Ende der 70er Jahre von Jean-Raymond Abrial erdacht und von der “Programming Research Group“ der Universität Oxford weiterentwickelt, in der Abrial ebenfalls mitarbeitete. Im Jahr 2002 wurde die Z-Notation durch die ISO standardisiert[7]. An Literatur, die sich mit der Z-Notation auseinandersetzt, mangelt es nicht. Gern genannt, um einen Einstieg zu erhalten, ist das Buch „Z Notation, A Reference Manual“ von J. M. Spivey [3], aus dem auch hier ein Auszug eines einfachen Beispiels gezeigt werden sollen.
Die Notation basiert auf der Mengenlehre und der Prädikatenlogik erster Stufe. Z wurde entworfen, um von Menschen gelesen zu werden, nicht unbedingt von Maschinen. Dazu passt das Konzept der Schemata, die es ermöglichen, eine Spezifikation zu strukturieren. Durch eine gute Strukturierung soll eine Präsentation in kleinen verständlichen Abschnitten erfolgen, wobei zusätzlich ein informeller Kommentar den Inhalt erläutern kann.

Einführendes Beispiel: „Geburtstagsbuch“ ein Geburtstagsverwaltungssystem (nach [3]).

Zu Beginn des Beispiels werden zwei Grundtypen oder Basistypen deklariert „*NAME*“ und „*GEBURTSDATUM*“. In einer Spezifikation in Z muss jedes Objekt einem Typ zugeordnet sein, wobei ein Typ in Z eine Menge darstellt. Dementsprechend steht „*NAME*“ in diesem Beispiel für die Menge aller Namen, „*GEBURTSDATUM*“ für die Menge aller Geburtsdaten. Basistypen sind Mengen, über deren innere Struktur nichts bekannt ist. Die in der Spezifikation verwendeten Typen müssen bekannt gemacht werden, bevor sie das erste Mal in einem Schema erscheinen. Eine Deklaration als Basistyp erfolgt in eckigen Klammern:

[6] Homepage: http://www.cafm.lsbu.ac.uk/
[7] Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics, ISO/IEC 13568:2002

[NAME, GEBURTSDATUM]

Typen können auch durch direktes Aufzählen ihrer Elemente wie folgt definiert werden:

NAME::= Jochen | Michael | Susanne

Diese präzise beschriebenen Mengen werden als „freie Typen“ bezeichnet.

Geburtstagsbuch

$personen: \mathbb{P}\, NAME$

$geburtstag: NAME \pfun GEBURTSDATUM$

$personen = \mathbf{dom}\ geburtstag$

Das Schema *Geburtstagsbuch* besteht aus zwei Teilen, wobei im oberen Teil Variablen deklariert werden (Deklarationsteil). Im unteren Teil werden die Beziehungen der deklarierten Objekte beschrieben (Prädikatteil).
Im Beispiel ist „*personen*“ eine Menge von Namen. „*geburtstag*“ ist definiert als eine partielle Abbildung von der Menge der Namen auf die Menge der Geburtsdaten. Mit der Bedingung „*personen* = dom *geburtstag*“ wird die Menge „*personen*“ beschränkt, wobei „dom *geburtstag*“ den Definitionsbereich der Funktion „*geburtstag*“ beschreibt. Mit dem Schema „*Geburtstagsbuch*“ werden die Zustände des Systems beschrieben. Ein möglicher Zustand des Systems ist:

$personen = \{Jochen, Michael, Susanne\}$
$geburtstag = \{Jochen \mapsto 25\text{-Mär}, Michael \mapsto 20\text{-Dez}, Susanne \mapsto 20\text{-Dez}\}$

Das Schema “*Geburtstag_hinzufügen*” beschreibt eine Operation des Systems, die eine Änderung des Zustandes bewirkt.

Geburtstag_hinzufügen

$\Delta\, Geburtstagsbuch$

$name? : NAME$

$datum? : GEBURTSDATUM$

$name? \notin personen$

$geburtstag' = geburtstag \cup \{ name? \mapsto datum? \}$

Die Deklaration „Δ*Geburtstagsbuch“* weist auf eine Änderung des Systemzustandes hin. Die „Δ“-Schreibweise ist eine verkürzte Variante von „*Geburtstagsbuch*“ und „*Geburtstagsbuch’* “. „*Geburtstagsbuch*“ steht für das obige Schema. „*Geburtstagsbuch’* “ entspricht ebenfalls diesem Schema mit dem Unterschied, dass jede Variable mit einem „’„ versehen ist. Diese Markierung weist darauf hin, dass es sich dabei um Variablen handelt, die den Folgezustand beschreiben. Weiterhin wird durch „Δ“ darauf aufmerksam gemacht, dass sich der Systemzustand ändert. Die deklarierten Variablen „*name?*“ und „*datum?*“ sind Eingabevariablen. Die Namen von Eingaben werden per Konvention mit einem Fragezeichen versehen, die von Ausgaben mit Ausrufungszeichen. Wann wird nun ein Geburtstag hinzugefügt? Wenn der neue Name noch nicht in der Menge „*personen*“ enthalten ist, gilt:
„*name?* $\notin$ *„peronen“*. Falls die Bedingung erfüllt ist, besteht die Menge der Geburtstage im Folgezustand aus den bisherigen Elementen vereint mit dem neuen Element $\{name? \nrightarrow datum?\}$.

Natürlich muss es auch möglich sein, einen Eintrag wieder zu entfernen.

Geburtstag_entfernen

Δ *Geburtstagsbuch*
name? : NAME
datum? : GEBURTSDATUM

$\{ name? \mapsto datum? \} \in geburtstag$
$geburtstag' = geburtstag \setminus \{ name? \mapsto datum? \}$

Wenn das Paar $\{name? \mapsto datum?\}$ in der Menge „*geburtstag*“ enthalten ist, so soll es entfernt werden und im Folgezustand nicht mehr enthalten sein.

Schließlich soll noch der Initialzustand des Systems beschrieben werden:

Init_Geburtstagsbuch

Geburtstagsbuch

$personen = \varnothing$

Hierzu ist anzumerken, dass der Initialzustand von Woodcock und Davis in „Using Z – Specification, Refinement, and Proof“ [4] mit der Begründung anders beschrieben wird, dass ein Initialzustand ein Zustand ohne Vorzustand ist. Diese Variante der Beschreibung des initialen Zustandes wird in den von mir erstellten Spezifikationen verwendet.

Das Schema „*Init_Geburtstagsbuch*“ Version 2:

Init_Geburtstagsbuch

$$Geburtstagsbuch'$$

$$personen' = \varnothing$$

An dieser Stelle möchte ich mit der Vorstellung der Z-Notation schließen. Die angesprochenen einfachen Formalismen sollten genügen, um die folgenden Spezifikationen zu verstehen. Für weitere Informationen zur Z-Notation verweise ich auf die Internetseite http://vl.zuser.org/ bzw. auf die Lehrbücher von Bowen [2], Spivey [3], Woodcock und Davis [10], Jacky [11] und Curry [12].

1.5 Zielstellung und Ergebnisse der Arbeit

Die Zielstellung dieser Arbeit, ist die Überprüfung und Präzisierung von informalen Spezifikationen eines realen Systems. Für diesen Zweck sollen formale Spezifikation des Systems in der Z-Notation entwickelt werden. Ausgewählt wurden dazu zwei typische Anwendungsfälle von XCTL. Neben der Erstellung der Spezifikationen sollen auch Aufwand und Nutzen betrachtet werden.
Im nächsten Abschnitt der Arbeit wird zunächst das Erstellen der formalen Spezifikationen der zwei Teilsysteme des XCTL-Programms betrachtet. Im dritten Abschnitt werden Fehler oder Probleme der vorhandenen verbalen Spezifikationen dargestellt, die während der Arbeit identifiziert werden konnten. Eine Zusammenfassung erfolgt in Abschnitt vier.
Als Ergebnisse dieser Arbeit sind zwei formale Verhaltensspezifikationen sowie eine erweiterte verbale Verhaltensbeschreibung erstellt worden. Die drei Dokumente sind im Anhang angegeben. Des Weiteren konnte gezeigt werden, dass eine Beschreibung des Systemverhaltens in der Z-Notation zu einer Verbesserung der Dokumentation beitragen kann.

2. Formale Spezifikation

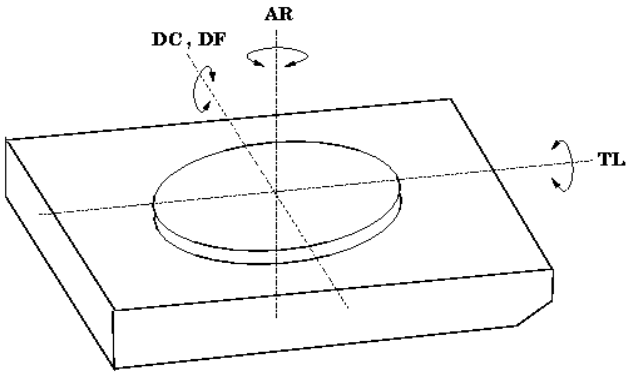
Im Folgenden soll die Erstellung der formalen Spezifikationen für das XCTL-System im Mittelpunkt stehen. Das erste Teilsystem, das spezifiziert werden soll, ist „Manuelle Justage (Alt)“, das zweite „0-dimensionale Detektoren, Teilfunktion Zählerfenster“. Die formalen Spezifikationen sind im Anhang der Arbeit vollständig angegeben.
Zu beiden Teilsystemen existiert eine verbale Beschreibung unter den Bezeichnungen: „Verhaltensspezifikation (Pflichtenheft) – XCTL – Steuerprogramm; Funktion: Probe und Kollimator manuell justieren“, in der Dokumentversion 2.2 (4.7.2001) [4] und „Verhaltensspezifikation XCTL-Steuerprogramm, Funktion: 0-dimensionale Detektoren, Teilfunktion: Zählerfenster“ in der Dokumentversion 2.04 (15.06.2006) [5].
In Abschnitt 2.3 soll zudem der Zeitaspekt betrachtet werden und wie dieser in die Modelle einbezogen werden kann.
Ausgehend von den verbalen Beschreibungen und dem Programm sollen Spezifikationen in der Z-Notation entstehen und die Vorgehensweise aufgezeigt werden. Vorab erfolgt eine Beschreibung der Funktionen.

2.1 Manuelle Justage (Alt)

Eine erste noch unvollständige Version der „Manuelle Justage (Alt)“ ist im Rahmen einer Studienarbeit entstanden [9], dabei wurden Erfahrungen mit der Z-Notation sowie mit dem Erstellen einer Spezifikation gesammelt. Die in [9] erstellte Version wurde einer kritischen Betrachtung unterzogen; Variablennamen wurden zum besseren Verständnis geändert und einige Schemata verändert. Ein Zeitmodell ist hinzugefügt worden. Die Einbeziehung der Zeit in das spezifizierte Verhalten ist jedoch relativ gering und könnte noch erweitert werden. Eine Idee hierzu wird in Abschnitt 2.3.2. ausgeführt.
Es ist fraglich, ob eine Verhaltensspezifikation eines „Ist-Zustandes“ als vollendet bezeichnet werden kann, da auch mit dieser Arbeit neue Details entdeckt wurden, die das Verhalten betreffen.

2.1.1 Beschreibung der Funktion

Die „Manuelle Justage (Alt)“ ist eine Funktion, mit der eine Probe durch direkte Steuerung von Antrieben in eine gewünschte Position gebracht werden kann. Dazu stehen vier verschiedene Antriebe zur Verfügung. Abbildung 3 zeigt eine schematische Darstellung.



AR – Azimutale Rotation
TL – Tilt, Verkippung
DC – Beugung (grob)
DF – Beugung (fein)

Abb. 3: Die vier Antriebe zur Positionierung der Probe

Ein fünfter Antrieb (CC, Kollimator Krümmung) hat Auswirkungen auf die Krümmung des Kollimators (Abb. 4). Dieser wird ebenfalls durch diese Funktion gesteuert.

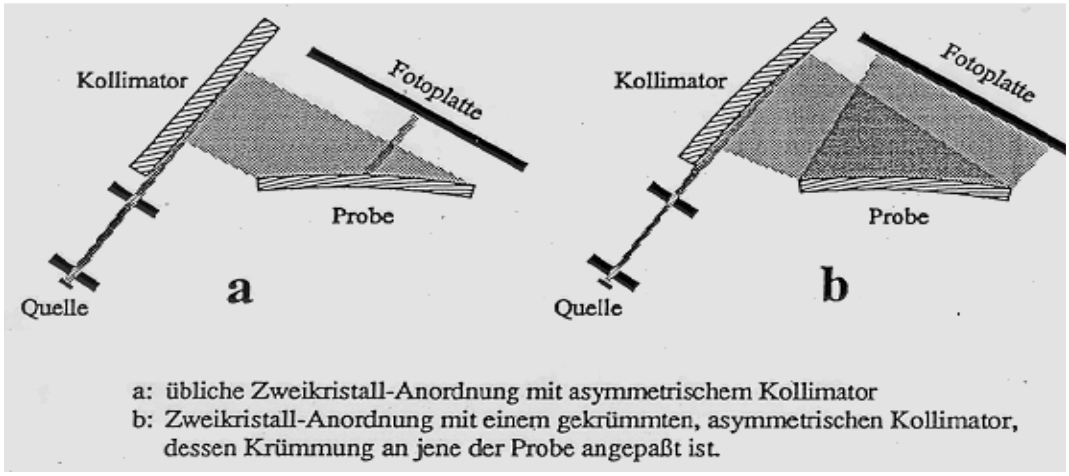


Abb. 4: Schema der Zweikristall-Topographie

Für die verschiedenen Antriebe stehen drei unterschiedliche Betriebsarten zur Verfügung:

- Der Direktbetrieb überführt einen Antrieb zu einer vom Nutzer eingegeben Position, soweit diese innerhalb eines antriebs-spezifischen Intervalls liegt.
- Der Schrittbetrieb führt eine Bewegung in einer festgelegten Schrittweite und in eine vom Nutzer bestimmte Richtung aus.
- Der Fahrbetrieb führt eine Bewegung in eine vom Nutzer bestimmte Richtung aus, solange dieser die Eingabe (Maustaste) betätigt.

In den zwei Betriebsarten Direktbetrieb und Fahrbetrieb bewegen sich die Antriebe mit einer Geschwindigkeit, die vom Nutzer festgelegt werden kann. Im Schrittbetrieb soll die Bewegung immer mit maximaler Geschwindigkeit durchgeführt werden. Die Schrittweite kann verändert werden. Für alle veränderbaren Parameter existieren Grenzwerte, wie zum Beispiel eine minimale und maximale Geschwindigkeit oder eine minimale und maximale Schrittweite. Diese Werte sind in einer Initialisierungsdatei definiert[8].
Die Positionsangaben für die Antriebe Beugung (fein), Beugung (grob) und Azimutale Rotation erfolgen in Winkelsekunden, für den Antrieb Tilt in Winkelminuten und für den Antrieb Kollimator Krümmung in Mikrometern. Abbildung 5 gibt die Dialogbox für die „Manuelle Justage (Alt)“ wieder, in der man die hier eingeführten Angaben wieder findet:

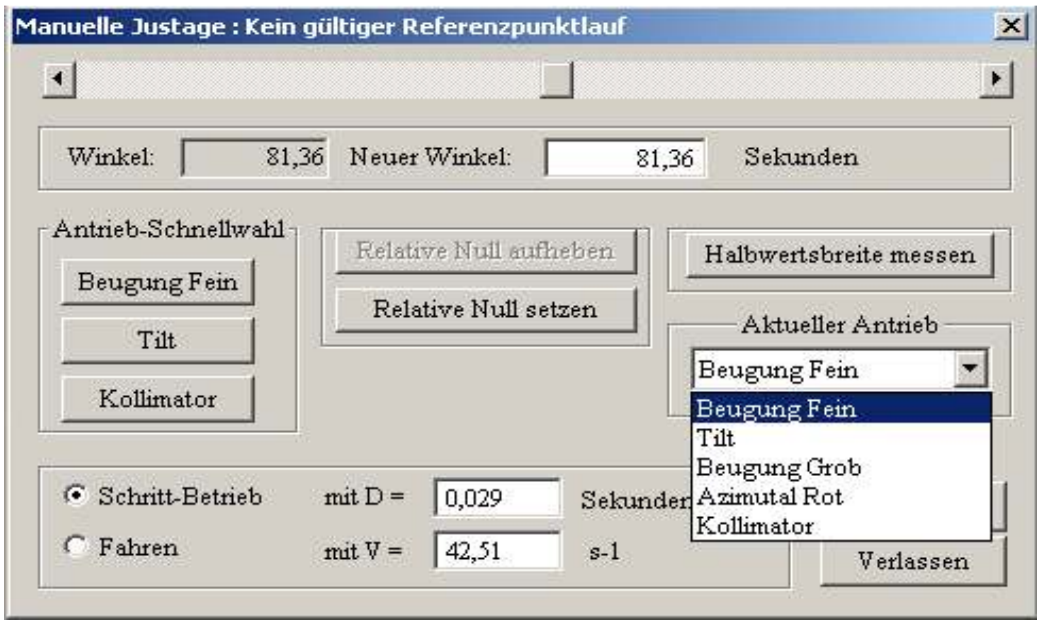


Abb. 5: Dialogbox für Manuelle Justage (Alt)

Die aktuelle Position des aktivierten Antriebes wird im oberen Bereich der Dialogbox dargestellt. Die Laufleiste zeigt die Position innerhalb des Bewegungsintervalls des Antriebes an. Im Feld ‚Winkel’ ist die aktuelle Position erkennbar. Die Auswahl der Antriebe kann mittels Schnellwahl erfolgen (links) oder per Auswahl aus dem Menü ‚Aktueller Antrieb’, in dem der aktive Antrieb angezeigt wird. Im unteren Bereich können die Betriebsarten gewählt sowie die Schrittweite und die Geschwindigkeit eingegeben werden. Eine Bewegung im Schritt - oder Fahrbetrieb wird durchgeführt, wenn das linke oder rechte Endelement der Laufleiste betätigt wird. Erfolgt eine Eingabe in das Feld ‚Neuer Winkel’ und wird die ‚Enter-Taste betätigt, so wird der

[8] Der Dateiname ist hardware.ini, diese befindet sich im Programmordner von XCTL

Direktbetrieb aktiviert. Der Knopf „Relative Null Setzen“, ermöglicht dem Nutzer einen Fixpunkt zu setzen: Die aktuelle Position bekommt als relative Position den Wert 0,0 zugeordnet. Der Balken der Laufleiste hilft bei der Orientierung. Ist dieser mittig positioniert, so befindet sich der jeweilige Antrieb auf seinem realen Nullpunkt, bei gesetzter relativer Null auf dem relativen Nullpunkt. Der reale Nullpunkt ist die Position in der Mitte des Bewegungsintervalls eines Antriebes.

2.1.2 Erstellung der Spezifikation

Die ersten Schritte auf dem Weg zur Z-Spezifikation waren das genaue Lesen der Verhaltensspezifikation und ein „Ausprobieren“ der Software. Das „Ausprobieren“ erfolgte zum einen, um die Software kennen zu lernen und zum anderen zum besseren Verständnis der in der Verhaltensspezifikation beschriebenen Funktionen. Nach einer gewissen Testzeit entstanden Verwirrungen. Viele der beschriebenen Eigenschaften trafen nicht ein und zudem kamen einige Merkwürdigkeiten zum Vorschein, die im Folgenden noch genauer betrachtet werden sollen. Das deutet bereits darauf hin, dass die verbale Spezifikation an einigen Stellen unpräzise, unvollständig und zum Teil inkorrekt gewesen ist. Die Software wurde ohne angeschlossenen Messplatz untersucht, das heißt die Bewegungen der zu steuernden Motoren wurden simuliert.

Die grundlegenden Informationen, die aus der Verhaltensspezifikation entnommen werden können, sind:

- Manuelle Justage (Alt) ist ein Teilsystem des XCTL-Steuerprogramms
- fünf verschiedene Antriebe können gesteuert werden
- für jeden Antrieb existieren Parameter wie Position, Geschwindigkeit, Schrittweite
- drei Betriebsarten stehen für jeden Antrieb zur Verfügung (nur zwischen Fahr- und Schrittbetrieb kann gewählt werden)
- die Bewegung der Antriebe ist unabhängig voneinander, Ausnahme: Beugung (grob) und Beugung (fein) können nicht gleichzeitig bewegt werden

Aus der Untersuchung der Software ergab sich, dass jedes Detail der verbalen Verhaltensspezifikation vor der Umsetzung in die Z-Notation mit dem Programm getestet werden sollte, eben deshalb, weil die existierende Verhaltensspezifikation oben erwähnte Mängel aufweist.
Als erster Schritt auf dem Weg zur formalen Spezifikation wird die Beschreibung des Systemzustandes vorgenommen. In erster Linie wird der

Systemzustand durch die aktuellen Positionen der Antriebe beschrieben. Hinzu kommen weitere nicht zu vernachlässigende Aspekte. So existieren für jeden Antrieb Parameter wie die aktuelle Geschwindigkeit, die aktuelle Schrittweite und die aktuelle Betriebsart. Ein Systemzustand kann demnach durch die:

- aktuelle Geschwindigkeit
- aktuelle Schrittweite
- aktuelle Betriebsart
- aktuelle Position

aller fünf Antriebe beschrieben werden.
Die Anfangsparameter zu den einzelnen Antrieben befinden sich in der Datei hardware.ini. Wird XCTL gestartet, werden die Parameter aus der Datei gelesen. Als nächsten Schritt muss der Nutzer die Funktion „Manuelle Justage (Alt)“ aufrufen.
Tests mit dem Programm ergaben, dass einige Parameter nur verfügbar sind, während „Manuelle Justage (Alt)“ aktiviert ist. Wieder andere bleiben verfügbar solange XCTL aktiv ist. Es stellte sich die Frage, ob die Initialisierungsdatei mit in das Modell eingebunden werden muss. Sie wurde mit ja beantwortet, da die Positionen der Antriebe bei Beendigung des Programms in der Datei gesichert werden. Das Schema *‚INI_File’* stellt eine Modellierung dieser Datei dar, in dem alle für das Teilsystem relevanten Variablen deklariert werden.
Im Systemmodel müssen Variablen für alle fünf Antriebe deklariert werden. Aus Gründen der Vereinfachung wurde für die numerischen Variablen der Datentyp $\mathbb{Z}$ gewählt.

Freie Typen

Zu Beginn werden freie Typen definiert, das sind Mengen deren Elemente direkt aufgezählt werden:

EREIGNISSE ::= *Links*
| *Enter*
| *Rechts*
| *Ausfuehren_Manuelle_Justage_Alt*
| *Verlassen_Manuelle_Justage*
| *Verlassen_XCTL*
| *Setze_Relative_Null*
| *Relative_Null_Aufheben*

Der Typ „*EREIGNISSE*“ soll vom Nutzer der Software getätigte Eingaben repräsentieren.
Für die Modellierung der Antriebe werden zwei Typen eingeführt:

$$ANTRIEBSART ::= fahrbetrieb \mid schrittbetrieb$$
$$ANTRIEB ::= DC \mid DF \mid AR \mid TL \mid CC$$

Der Typ Status wird in der Spezifikation genutzt, um beispielsweise zu zeigen, ob die Funktion Manuelle Justage (Alt) aktiv ist.

$$STATUS ::= Ein \mid Aus$$

Ini-File

Das erste modellierte Schema ist „*INI_File*“.

INI_File

$dc\_ist\_position, dc\_position\_min, dc\_position\_max, dc\_schrittweite,$
$dc\_schrittweite\_min, dc\_schrittweite\_max, dc\_geschwindigkeit,$
$dc\_geschwindigkeit\_max: \mathbb{Z}$
$df\_ist\_position, df\_position\_min, df\_position\_max, df\_schrittweite,$
$df\_schrittweite\_min, df\_schrittweite\_max, df\_geschwindigkeit,$
$df\_geschwindigkeit\_max: \mathbb{Z}$
$ar\_ist\_position, ar\_position\_min, ar\_position\_max, ar\_schrittweite,$
$ar\_schrittweite\_min, ar\_schrittweite\_max, ar\_geschwindigkeit,$
$ar\_geschwindigkeit\_max: \mathbb{Z}$
$tl\_ist\_position, tl\_position\_min, tl\_position\_max, tl\_schrittweite,$
$tl\_schrittweite\_min, tl\_schrittweite\_max, tl\_geschwindigkeit,$
$tl\_geschwindigkeit\_max: \mathbb{Z}$
$cc\_ist\_position, cc\_position\_min, cc\_position\_max, cc\_schrittweite,$
$cc\_schrittweite\_min, cc\_schrittweite\_max, cc\_geschwindigkeit,$
$cc\_geschwindigkeit\_max: \mathbb{Z}$

Alle Parameter, die in der Datei hardware.ini vorkommen und für die Teilfunktion relevant sind, werden in dem Schema *INI_File* als Variablen deklariert. Beschränkungen existieren nicht, so dass kein Prädikatteil erscheint.

XCTL

Im Schema XCTL und den nachfolgenden Schemata wird der Einfachheit und Übersichtlichkeit halber immer nur ein Motor aufgenommen (als Beispiel der Motor DC - Bewegung grob). Die vollständige Fassung findet sich im Anhang A. Dort werden Variablen aller fünf Antriebe deklariert.

Im Schema *XCTL* werden alle Variablen deklariert, die verfügbar sind, wenn das XCTL-Programm aktiv ist.

XCTL

$$
\begin{array}{l}
INI\_File \\
dc\_rel\_null,\ dc\_rel\_ist\_position, \\
dc\_rel\_position\_min,\ dc\_rel\_position\_max: \mathbb{Z} \\
aktueller\_antrieb: ANTRIEB \\
manuelle\_justage: STATUS \\
\hline
dc\_rel\_position\_min = dc\_position\_min - dc\_rel\_null \\
dc\_rel\_position\_max = dc\_position\_max - dc\_rel\_null
\end{array}
$$

Beschränkungen gibt es hier für die minimale und die maximale relative Position des Antriebes DC, dargestellt durch die Variablen *dc_rel_position_min* und *dc_rel_position_max*. Die minimale relative Position des Antriebes muss gleich sein mit der realen minimalen Position abzüglich eines Offsets (der Wert der relativen Null des Antriebes DC). Entsprechendes gilt für die maximale relative Position.

Manuelle Justage

Das Schema *Manuell_Justage* umfasst XCTL und beschreibt den aktuellen Zustand der Manuellen Justage.

Manuelle_Justage

$$
\begin{array}{l}
XCTL \\
dc\_schrittweite\_aktuell,\ dc\_geschwindigkeit\_aktuell, \\
dc\_geschwindigkeit\_aktuell\_2: \mathbb{Z} \\
aktuelle\_antriebsart\_dc: ANTRIEBSART
\end{array}
$$

Im Schema *Manuelle_Justage* werden zwei Variablen für eine aktuelle Geschwindigkeit vorgestellt. Die zweite Variable wurde für die Beschreibung benötigt, da beim Wechsel der Antriebsart von Schrittbetrieb zu Fahrbetrieb oder umgekehrt eine Veränderung der aktuellen Geschwindigkeit vorgenommen wird.
Die Zuordnung der Variablen zu den Schemata *XCTL* und *Manuelle_Justage* erfolgte nach Überprüfung am Programm. Wird beispielsweise der Wert der Schrittweite verändert und anschließend die Funktion Manuelle Justage beendet und wieder neu gestartet, so wird der veränderte Wert nicht angezeigt, sondern ein Standardwert. Man kann also davon ausgehen, dass die Variable für die aktuelle Schrittweite nur solange existiert, wie das Teilsystem aktiv ist. Aus diesem kleinen Beispiel wird deutlich, dass das Verlassen oder Beenden von Manuelle Justage ebenfalls eine Zustandsänderung nach sich zieht - die Änderung der aktuellen Schrittweite. Daher wird eine Operation modelliert werden, die diesen Aspekt berücksichtigt.
Die Systemzustände werden mit den drei Schemata *INI_File, XCTL* und *Manuelle_Justage* beschrieben.
Zusätzlich wurden die Schemata *Init_XCTL* und *Init_Manuelle_Justage* definiert, um den initialen Systemzustand zu beschreiben.

Init_XCTL

In diesem Schema wird beschrieben, welchen Werte Variablen nach dem Start von XCTL besitzen.

Init_XCTL

INI_File
XCTL'

---

dc_rel_null' = 0
dc_rel_ist_position' = *dc_ist_position*
aktueller_antrieb' = *DF*
manuelle_justage' = *Aus*

Die „Relative Null“ des Motors DC ist nach dem Start von XCTL gleich Null. Die relative Ist-Position des Motors DC besitzt den Wert der absoluten Position des Motors DC, der in der Initialisierungsdatei zu finden ist. Der aktuelle Antrieb des Systems ist zu Beginn Beugung fein (DF) und die „Manuelle Justage“ ist nicht aktiv.

Init_Manuelle_Justage

Im Startzustand der „Manuellen Justage“ besitzen die Variablen die hier beschriebenen Werte.

Init_Manuelle_Justage

$$XCTL$$
$$Manuelle\_Justage'$$

$$dc\_schrittweite\_aktuell' = dc\_schrittweite$$
$$dc\_geschwindigkeit\_aktuell' = dc\_geschwindigkeit\_max$$
$$dc\_geschwindigkeit\_aktuell\_2' = dc\_geschwindigkeit\_max$$
$$aktuelle\_antriebsart\_dc' = schrittbetrieb$$

Die Werte für die Schrittweite und der Geschwindigkeit werden der Initialisierungsdatei entnommen. Die aktuell ausgewählte Antriebsart ist der Schrittbetrieb.

Operationen

Nachdem die Systemzustände beschrieben wurden, ergibt sich die nächste Frage: Wie oder durch welche Operationen wird der Zustand des Systems verändert?

Durch folgende Ereignisse ändert sich der Zustand des Systems:

- die Bewegung der Antriebe (Direktbetrieb, Schrittbetrieb, Fahrbetrieb)
- den Wechsel der Antriebe
- den Wechsel der Antriebsart
- das Setzen / Aufheben der relativen Null
- die Änderung der Schrittweite und der Geschwindigkeit
- das Beenden von Manuelle Justage (Alt), XCTL

Ausgehend von dem Zustandsmodell ändert sich durch jede dieser Operationen der Zustand. Zum Beispiel wird durch eine Bewegung im Direktbetrieb die aktuelle Position des aktuellen Antriebes verändert. Nachfolgend sind einige Beispiele für die Umsetzung der Operationen in die Z-Notation aufgeführt.

Bewegung des Antriebs DC im Direktbetrieb

Ausgangspunkt ist die entsprechende Beschreibung aus der verbalen Verhaltensspezifikation:

2.3 c) 1. *"... Die Soll-Position wird im Eingabefeld Neuer Winkel als Absolutwert angegeben. Der Antrieb fährt in die gewünschte Soll-Position. Die Bewegung wird (derzeit) durch Drücken der 'Enter'-Taste ausgelöst…"*

2.3 d)
„*…In den Betriebsarten Direktbetrieb und Fahrbetrieb kann die Bewegungsgeschwindigkeit des Antriebs vorgegeben werden. Sie kann vom Benutzer im Eingabefeld Fahren mit*
V = ... s-1 eingetragen werden…".
2.3.e)1. *„Unzulässige Sollposition: Wird im Direktbetrieb eine Soll-Position eingegeben, die außerhalb des zulässigen Wertebereichs liegt, so wird die Positionsangabe auf die minimal oder maximal zulässige Position korrigiert, je nachdem, ob der zulässige Wertebereich unterschritten oder überschritten wird. ..."*

Aus den Beschreibungen ist gut ersichtlich wie der Direktbetrieb arbeitet. Für die Modellierung ergab sich, dass ein Typ „EREIGNISSE" definiert wurde, dessen Werte die Eingaben der Nutzer repräsentieren sollen. Im Fall des Direktbetriebes war es das Drücken der ‚Enter-Taste. Um den Motor in Bewegung zu setzen, ist zusätzlich die Eingabe einer neuen Position nötig. Man muss eine Eingabevariable definieren: *neuer_winkel?* vom Typ $\mathbb{Z}$.

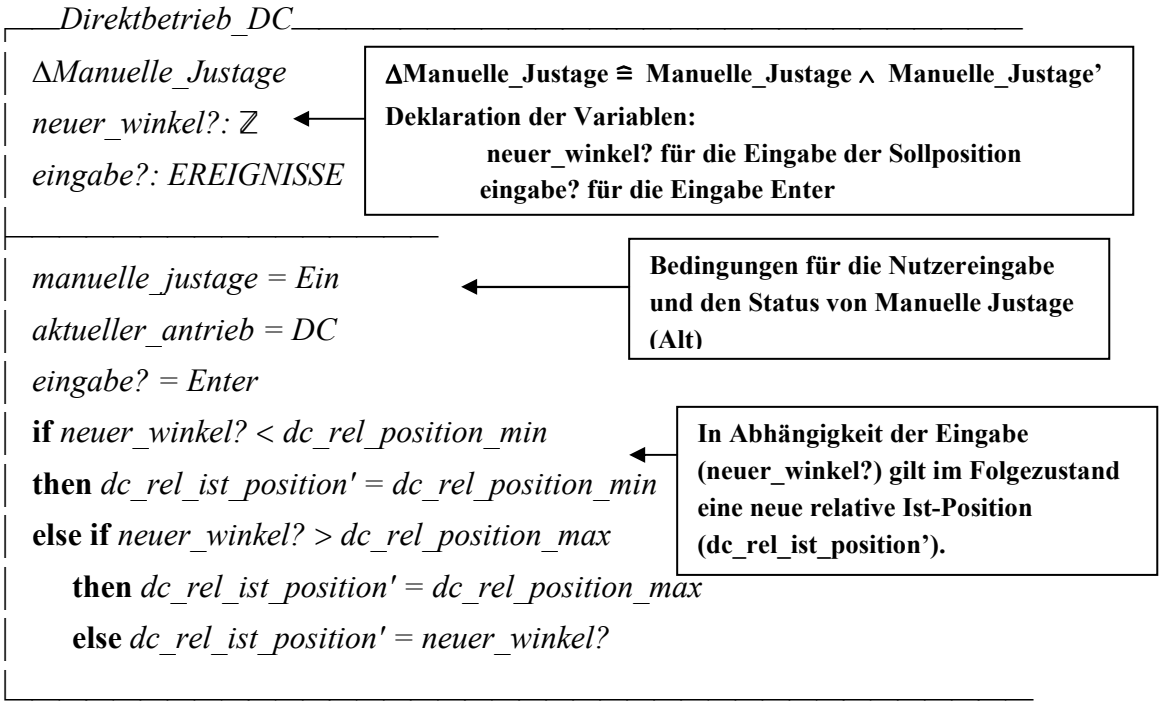
Die Bewegung des Motors verursacht eine Zustandsänderung und die aktuelle Position des Antriebes DC ändert sich. In diesem Fall liegt eine Veränderung des Schemas *Manuelle_Justage* vor.

Welche Bedingungen müssen eingehalten werden?

- Die Motorposition im Folgezustand (*dc_rel_ist_position'*) soll gleich der eingegebenen Position sein (*neuer_winkel?*) und muss innerhalb der erlaubten Positionsbeschränkungen liegen.
- Der Nutzer muss die Eingabe mit ‚Enter' bestätigen.
- Der aktuelle Antrieb muss DC sein.
- Die „Manuelle Justage" muss aktiviert sein.

Der Status von Manuelle Justage (Alt) wurde mit in das Modell aufgenommen (Statusvariable manuelle_justage), da eine Steuerung nur möglich ist, wenn die

Funktion Manuelle Justage (Alt) vom Nutzer im XCTL-Programm gestartet wird und aktiv bleibt.



Für alle anderen Antriebe muss ebenfalls ein solches Schema erstellt werden oder man nutzt die Möglichkeit des „renaming“. Das Ziel wird erreicht, indem die relevanten Variablen durch die eines anderen Antriebes ersetzt werden.

Im Beispiel für den Antrieb DF:

Direktbetrieb_DF ≙ *Direktbetrieb_DC*[
df_geschwindigkeit_aktuell/*dc_geschwindigkeit_aktuell*,
df_rel_position_min/*dc_rel_position_min*,
df_rel_position_max/*dc_rel_position_max*,
df_rel_istposition/*dc_rel_ist_position*]

Das Prinzip: Schema_neu ≙ Schema_alt [variable_neu / variable_alt,…]

Wechsel der Antriebe

Um einen anderen als den aktuellen Antrieb (Motor) zu steuern, muss der Nutzer einen Wechsel des Antriebes vornehmen. Das Schema *Auswahl_Antrieb* beschreibt diese Operation:

Auswahl_Antrieb

$\Delta XCTL$

neuer_antrieb?: ANTRIEB

---

manuelle_justage = Ein

aktueller_antrieb' = neuer_antrieb?

Der Nutzer muss per Mausklick auf das Popup-Menü oder per direkter Auswahl in der Dialogbox einen neuen Antrieb wählen.

Wechsel der Antriebsart für den Antrieb DC

Eine weitere Operation ist der Wechsel der Antriebsart. Hier kann der Nutzer zwischen Fahrbetrieb und Schrittbetrieb wählen. Der Direktbetrieb muss nicht betrachtet werden, da er unabhängig von den beiden anderen Antriebsarten immer verfügbar ist. Das Programm verhält sich jedoch in Bezug auf die aktuelle Geschwindigkeit nicht nach den Erwartungen des Nutzers.

Ausgangspunkt sei wieder die verbale Verhaltensspezifikation:

2.3 c „*Bewegungsgeschwindigkeit:*

In den Betriebsarten Direktbetrieb und Fahrbetrieb kann die Bewegungsgeschwindigkeit des Antriebs vorgegeben werden. Sie kann vom Benutzer im Eingabefeld Fahren mit V = ... s-1 eingetragen werden. Im Schrittbetrieb wird der Antrieb immer mit der maximal möglichen Geschwindigkeit bewegt (hier liegt der Gedanke zu Grunde, dass in sehr kleinen Schritten bewegt wird, die mehrfach wiederholt werden sollen).“

Der Wechsel der Antriebsart wird in der verbalen Beschreibung nicht explizit betrachtet. Bei diesem Wechsel wird die aktuelle Geschwindigkeit verändert. Eine nähere Erläuterung dazu erfolgt in Abschnitt 3.1 c). Das folgende Operationsschema beruht auf den Tests der Software:

Auswahl_Antriebsart_DC

$\Delta Manuelle\_Justage$

neue_antriebsart?: ANTRIEBSART

---

manuelle_justage = Ein

aktueller_antrieb = DC
aktuelle_antriebsart_dc' = neue_antriebsart?
if *aktuelle_antriebsart_dc = fahrbetrieb*
∧ *neue_antriebsart? = schrittbetrieb*
then *dc_geschwindigkeit_aktuell' = dc_geschwindigkeit_max*
if *aktuelle_antriebsart_dc = schrittbetrieb*
∧ *neue_antriebsart? = fahrbetrieb*
then *dc_geschwindigkeit_aktuell' = dc_geschwindigkeit_aktuell_2*

Erfolgt ein Wechsel der Antriebsart so wirkt sich dies auf die aktuelle Geschwindigkeit aus.

Bewegung des Antriebs DC im Schrittbetrieb

Eine Bewegung im Schrittbetrieb wird ausgelöst, wenn der Nutzer die Eingabe „Links“ oder „Rechts“ vornimmt. Das geschieht durch das Betätigen der Pfeiltasten oder durch einen Mausklick auf das entsprechende Endelement der Bildlaufleiste in der Dialogbox von „Manuelle Justage (Alt)“ (Abb. 5). Voraussetzungen sind, dass der Schrittbetrieb die aktuell gewählte Antriebsart ist, dass die „Manuelle Justage“ aktiv ist und der aktuell gewählte Antrieb DC ist.

Schrittbetrieb_DC
ΔManuelle_Justage
eingabe?: EREIGNISSE
---
manuelle_justage = Ein
aktueller_antrieb = DC
aktuelle_antriebsart_dc = schrittbetrieb
if *eingabe? = Links*
∧ *dc_rel_ist_position - dc_schrittweite_aktuell ⩾ dc_rel_position_min*
then *dc_rel_ist_position' = dc_rel_ist_position - dc_schrittweite_aktuell*
if *eingabe? = Rechts*
∧ *dc_rel_ist_position + dc_schrittweite_aktuell ⩽ dc_rel_position_max*
then *dc_rel_ist_position' = dc_rel_ist_position + dc_schrittweite_aktuell*

In Abhängigkeit der Eingabe wird eine Bewegung in die gewählte Richtung ausgeführt. Im Fall der Eingabe „Links“ wird vom Wert der aktuellen relativen Ist-Position die aktuelle Schrittweite abgezogen, wenn dadurch die minimal erreichbare Position nicht überschritten würde. Der Antrieb befindet sich danach auf einer neuen Position. Im Fall „Rechts“ erfolgt die Bewegung analog, nur dass in diesem Fall die maximal erreichbare Position nicht überschritten werden darf. Würde durch einen Schritt eine Grenze überschritten werden, reagiert das Programm nicht.

Bewegung des Antriebs DC im Fahrbetrieb

Das Schema zur Beschreibung des Fahrbetriebs für den Antrieb DC ist relativ umfangreich. Neben den Beschränkungen für die Positionen wird hier ein zeitlicher Aspekt modelliert. Eine nähere Beschreibung des Zeitaspektes erfolgt in Abschnitt 2.3.

Fahrbetrieb_DC

Δ*Manuelle_Justage*
eingabe?: EREIGNISSE
laenge_der_eingabe: INTERVAL

---

manuelle_justage = Ein
aktueller_antrieb = DC
aktuelle_antriebsart_dc = fahrbetrieb
if *eingabe? = Links* $\wedge$ *dc_geschwindigkeit_aktuell* $>$ 0
$\wedge$ *dc_rel_ist_position*
*- dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
$<$ *dc_rel_position_min*
then *dc_rel_ist_position' = dc_rel_position_min*
else *dc_rel_ist_position'*
= dc_rel_ist_position
*- dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
$\vee$ (**if** *eingabe? = Recht* $\wedge$ *dc_geschwindigkeit_aktuell* $>$ 0
$\wedge$ *dc_rel_ist_position*
*+ dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
$>$ *dc_rel_position_max*
then *dc_rel_ist_position' = dc_rel_position_max*
else *dc_rel_ist_position'*

$$
\begin{array}{l}
\quad = dc\_rel\_ist\_position \\
\quad\quad + dc\_geschwindigkeit\_aktuell * period\ laenge\_der\_eingabe) \\
\quad \vee\ dc\_geschwindigkeit\_aktuell = 0 \\
\quad\ \wedge\ eingabe? = Links \\
\quad\ \wedge\ dc\_rel\_ist\_position' = dc\_rel\_position\_min \\
\quad \vee\ dc\_geschwindigkeit\_aktuell = 0 \\
\quad\ \wedge\ eingabe? = Rechts \\
\quad\ \wedge\ dc\_rel\_ist\_position' = dc\_rel\_position\_max
\end{array}
$$

In Abhängigkeit der Eingabe „Links“ oder „Rechts“ erfolgt eine Bewegung in die jeweilige Richtung. Die Bewegung dauert so lange, wie diese Eingabe betätigt wird, wobei die Positionsgrenzen nicht überschritten werden dürfen. Ein besonderes Verhalten zeigt das Programm, wenn der Wert der aktuellen Geschwindigkeit gleich Null ist. Je nach Eingabe ergibt sich als Position im Folgezustand das Minimum oder Maximum.

Setzen der relativen Null für den Antrieb DC

Das folgende Schema stellt die Operation für das Setzen der relativen Null dar:

Relative_Null_Setzen_DC

$$
\begin{array}{l}
\Delta XCTL \\
eingabe?: EREIGNISSE \\
\hline
manuelle\_justage = Ein \\
aktueller\_antrieb = DC \\
eingabe? = Setze\_Relative\_Null \\
dc\_rel\_null' = dc\_rel\_ist\_position + dc\_rel\_null \\
dc\_rel\_ist\_position' = 0
\end{array}
$$

Betätigt der Nutzer das Feld „Relative Null setzen“ wird die aktuelle Position auf Null gesetzt. Die Variable *dc_rel_null* erhält als Wert die Summe aus der aktuellen Position des aktuellen Antriebs (in diesem Fall DC) und der ‚alten’ relativen Null dieses Antriebs.

Beenden von Manuelle Justage

Das folgende Schema beschreibt die Auswirkungen bei Beenden der Manuellen Justage (Alt). Wird die Dialogbox (z.B. Mausklick auf „Verlassen“) geschlossen, wird nicht nur die Manuelle Justage (Alt) beendet, sondern es werden auch alle sich in Bewegung befindlichen Antriebe gestoppt. Die aktuelle Position der Antriebe ist jedoch weiterhin im XCTL-Programm verfügbar. Die aktuellen Positionswerte werden demnach von XCTL gesichert.

Manuelle_Justage_Verlassen

$\Delta XCTL$

eingabe?: EREIGNISSE

dc_antrieb_ist_position?, df_antrieb_ist_position?, ar_antrieb_ist_position?,
tl_antrieb_ist_position?, cc_antrieb_ist_position?: $\mathbb{Z}$

---

eingabe? = Verlassen_Manuelle_Justage

dc_rel_ist_position' = dc_antrieb_ist_position? - dc_rel_null

Die relativen Positionen der Antriebe sind weiterhin verfügbar. Mit der Variablen *dc_antrieb_ist_position?* soll ein Input des Systems modelliert werden, der die aktuelle Position des Antriebes an XCTL für den Fall übergibt, dass die Funktion Manuelle Justage (Alt) geschlossen wird.

Die weiteren identifizierten Operationen sind: „Aufheben der relativen Null“, „Änderung der Schrittweite“, „Änderung der Geschwindigkeit“ für jeden Antrieb und „Verlassen von XCTL“. Sie wurden nach demselben Prinzip modelliert. Aufgrund der häufigen Tests und des unerwarteten Verhaltens des Programms war der Zeitaufwand für diese Modellierung hoch. Um jedoch den „Ist-Zustand“ der Teilfunktion „Manuelle Justage (Alt)“ so korrekt wie möglich darzustellen, waren sie gleichwohl unverzichtbar.

Systemoperationen

Der nächste Schritt ist die Zusammenfassung verschiedener Operationen zu einer Gesamtoperation oder Systemoperation. Die Steuerung des Antriebes DC könnte als Operation definiert werden. Sie setzt sich aus allen Operationen, die den Antrieb DC betreffen, zusammen.

$$
\begin{aligned}
&Steuerung\_DC \mathrel{\widehat{=}} \\
&Auswahl\_Antriebsart\_DC \\
&\lor\ Eingabe\_Schrittweite\_DC \\
&\lor\ Eingabe\_Geschwindigkeit\_DC \\
&\lor\ Direktbetrieb\_DC \\
&\lor\ Schrittbetrieb\_DC \\
&\lor\ Fahrbetrieb\_DC \\
&\lor\ Relative\_Null\_Setzen\_DC \\
&\lor\ Aufheben\_Relative\_Null\_DC
\end{aligned}
$$

Die Operationen der Antriebe können zusammengefasst werden und ergeben mit den übrigen Schemata Steuerung_Manuelle_Justage:

$$
\begin{aligned}
&Steuerung\_Manuelle\_Justage \mathrel{\widehat{=}} \\
&Auswahl\_Motor \\
&\lor\ Motor\_DC \land Steuerung\_DC \land \neg\ Steuerung\_DF \\
&\lor\ Motor\_DF \land Steuerung\_DF \land \neg\ Steuerung\_DC \\
&\lor\ Motor\_AR \land Steuerung\_AR \\
&\lor\ Motor\_TL \land Steuerung\_TL \\
&\lor\ Motor\_CC \land Steuerung\_CC \\
&\lor\ Manuelle\_Justage\_Verlassen
\end{aligned}
$$

Die vollständige Spezifikation ist im Anhang hinterlegt. Die obigen Beispiele sollen einen Einblick in die Vorgehensweise bei der Erstellung der Beschreibung der Teilfunktion „Manuelle Justage (Alt)“ in der Z-Notation vermitteln.

2.2 0-dimensionale Detektoren, Teilfunktion Zählerfenster

2.2.1 Beschreibung der Funktion

Die 0-dimensionalen Detektoren zählen die innerhalb eines Zeitintervalls einfallenden Photonen. Die Anzahl wird fortlaufend in einem Zählerfenster ausgegeben (siehe Abb.: 6). Zusätzlich kann die Zählrate akustisch ausgegeben werden; jedoch nur bei real existierenden angeschlossenen Detektoren.

Abb. 6: Das Zählerfenster

Das Zählerfenster öffnet sich direkt mit dem Programmstart. Um den automatischen Start zu verhindern, besteht jedoch die Möglichkeit einen Startparameter zu verändern.
Per Nutzereingabe über die Hauptmenüfunktion „Detektoren > Detektoren“ kann das Zählerkonfigurationsfenster (Abb. 7) geöffnet werden.

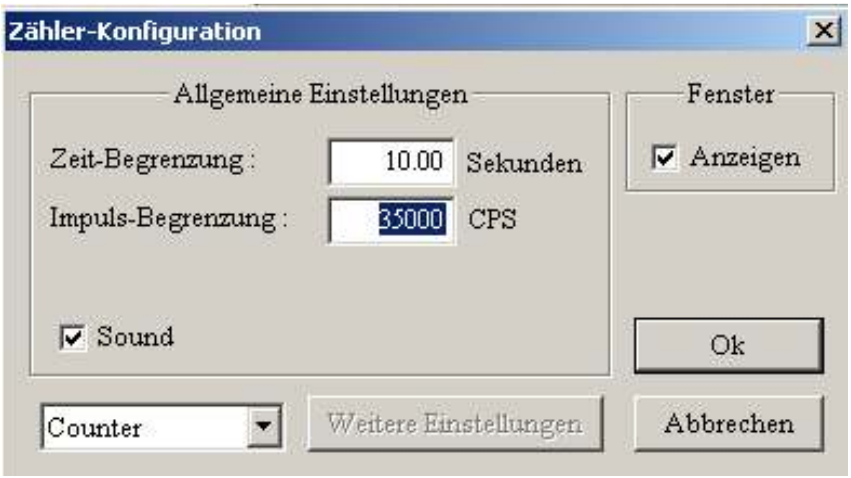


Abb. 7: Dialogbox Zählerfensterkonfiguration

Im Bereich „Allgemeine Einstellungen“ kann eine Zeit- und eine Impulsbegrenzung eingegeben werden. Die Zeitbegrenzung gibt die Länge des Intervalls an, in dem die einfallenden Photonen gezählt werden. Wird innerhalb dieses Zeitintervalls eine Photonenzahl erreicht, die der Impulsbegrenzung entspricht, startet der Zählvorgang von neuem. Die Impulsbegrenzung ist ein zweites Abbruchkriterium für die Zählung. Die akustische und die visuelle Ausgabe der Zählrate werden über die Felder „Sound“ bzw. „Anzeige“ gesteuert. Ist kein Häkchen im Feld Anzeige zu sehen, wird das Fenster aus Abbildung 7 ausgeblendet. Im linken unteren Bereich erfolgt die Auswahl des Detektors. Wird ein neuer Detektor gewählt, wird der Zählvorgang mit dem gewählten Detektor gestartet. Die Auswahl eines neuen Detektors, die Anzeige- und die Soundeinstellung werden unmittelbar wirksam, wohingegen eine Änderung der Zeit- und / oder der Impulsbegrenzung mit „OK“ oder „Enter“ bestätigt werden muss. Über ein „Popup“-Menü, zu öffnen per Mausklick (rechts) auf dem Zählerfenster, kann der Nutzer weitere Einstellungen vornehmen.

(Abb. 8: Zählerfenster mit „Pop-Up“-Menü)

Abbildung 8 zeigt das Menü. Gewählt werden kann zwischen einer digitalen Anzeige (momentan aktiv und im Hintergrund zu sehen) und einer Balkendarstellung. Über „Einstellungen Anzeige…“ erscheint ein Fenster, in dem Parameter für die Balkendarstellung gesetzt werden können. Der Punkt „Einstellungen Gerät…“ öffnet das Zähler-Konfigurationsfenster (Abb. 7). Mit „Log-Datei“ besteht die Option Messdaten in einer Datei zu sichern.

2.2.2 Erstellung der Spezifikation

Die grundlegende Idee bei der Modellierung ähnelt der von „Manuelle Justage (Alt)“. Das bedeutet, dass ebenfalls die Schemata *INI_Datei* und *XCTL* existieren. Alle relevanten Variablen, die aus Initialisierungsdateien gelesen

werden, sind in dem Schema *INI_Datei* deklariert. Sämtliche numerische Typen des Programms werden, ohne Berücksichtigung der tatsächlichen Typen, als natürliche Zahlen im Modell dargestellt. Für die Variablen, die Werte von Zeitbegrenzungen beinhalten, wurde der Typ *HUNDERTSTEL* gewählt, der aber dem Typ der natürlichen Zahlen entspricht. Dieser ist Bestandteil eines Zeitmodells, das in Abschnitt 2.3.3 beschrieben wird.

Am Anfang steht wieder die Definition freier Typen:

Freie Typen

Die Menge aller Detektoren ist gegeben durch:

DETEKTOR ::= detektor ⟪ℕ⟫
Die Elemente sind ein, aus dem Wort ‚detektor' und einer natürlichen Zahl, zusammengesetztes Paar. (Beispiele: detektor 0, detektor 5, detektor 100)

Die einzelnen in XCTL verfügbaren Detektortypen werden wie folgt definiert:

DETEKTORTYP ::= *RADICON* | *GENERIC* | *BRAUN_PSD* | *Test*
| *Simulant* | *PSD*

Der Typ Status wird in der Spezifikation genutzt, um beispielsweise zu zeigen, ob ein Fenster aktiv ist.

STATUS ::= *Ein* | *Aus*

Mit der folgenden Definition werden die einzelnen Start-up-Parameter dargestellt:

XCONTROLSTARTUP ::= *Startup_Nothing*
| *Startup*
| *Startup_Scan*
| *Startup_AreaScan*
| *Startup_Device*
| *Startup_ManualAdjustment*
| *NIL*

Die verschiedenen Anzeigemodi werden modelliert durch den Typ Anzeige:

$ANZEIGE ::= Digital \mid Balken$

Der Typ Ereignisse stellt die möglichen Eingaben des Nutzers dar:

$$
\begin{aligned}
EREIGNISSE ::= \; & Verlassen\_Zaehler\_Konfiguration \\
\mid \; & Verlassen\_XCTL \\
\mid \; & Verlassen\_Zaehler\_Fenster \\
\mid \; & Abbrechen \\
\mid \; & OK \\
\mid \; & Markierung\_Sound \\
\mid \; & Markierung\_Digitale\_Anzeige \\
\mid \; & Markierung\_Balken\_Darstellung \\
\mid \; & Markierung\_Fenster \\
\mid \; & Markierung\_Logarithmische\_Dartsellung \\
\mid \; & Markierung\_Log\_Datei \\
\mid \; & Auswahl\_Detektoren\_Detektoren \\
\mid \; & Auswahl\_Einstellungen\_Geraet \\
\mid \; & Auswahl\_Einstellungen\_Anzeige \\
\mid \; & Auswahl\_Detektor \\
\mid \; & Schließen \\
\mid \; & Enter
\end{aligned}
$$

Beschreibung des Systemzustands

Die Systemzustände werden durch die Schemata *INI_File, XCTL, Anzeige_Zaehlerfenster* und *Zaehler* beschrieben.

INI_File

Das Schema *Ini_File* beschreibt alle Variablen die in den Initialisierungsdateien vorkommen:

INI_File

$Detektoren: \mathbb{P}\, DETEKTOR$

$Zeitbegrenzungen\_Default: \mathbb{P}\, (DETEKTOR \times HUNDERTSTEL)$

Impulsbegrenzungen_Default: $\mathbb{P}$ (*DETEKTOR* $\times$ $\mathbb{N}$)

Soundeinstellung_Default: $\mathbb{P}$ (*DETEKTOR* $\times$ *STATUS*)

Detektortypen: $\mathbb{P}$ (*DETEKTOR* $\times$ *DETEKTORTYP*)

xcontrol_startup: XCONTROLSTARTUP

default_balkenanzahl: $\mathbb{N}$

min_zeitbegrenzung, max_zeitbegrenzung: HUNDERTSTEL

min_impulsbegrenzung, max_impulsbegrenzung: $\mathbb{N}$

min_intensitaetsbegrenzung, max_intensitaetsbegrenzung: $\mathbb{N}$

In den Initialisierungsdateien werden Parameter für Impulsbegrenzungen, Zeitbegrenzungen und Soundeinstellungen für jeden Detektor angegeben. Diese sind hier als Menge geordneter Paare definiert.

XCTL

In dem Schema werden alle Variablen, die bei aktivem XCTL-Programm verfügbar sind dargestellt.

XCTL

INI_File

zaehlerfenster, konfigurationsfenster: STATUS

Aktive_Detektoren: $\mathbb{P}$ *DETEKTOR*

sound: STATUS

anzeige: ANZEIGE

aktueller_detektor: DETEKTOR

aktuelle_zeitbegrenzung, sicherung_zeitbegrenzung: HUNDERTSTEL

aktuelle_impulsbegrenzung: $\mathbb{N}$

Die Menge *Aktive_Detektoren* beinhaltet alle Detektoren, die von XCTL erkannt werden. Die Variable *sicherung_zeitbegrenzung* wird benötigt, um die Änderung der aktuellen Zeitbegrenzung bei impulsbedingtem Neustart zu sichern.

Zählerfenster

Das Schema *Anzeige_Zaehlerfenster* beinhaltet alle Variablen die verfügbar sind, solange das Zählerfenster aktiv ist.

Anzeige_Zaehlerfenster

XCTL

aktuelle_intensitaet, default_intensitaet, min_intensitaet, max_intensitaet,
aktuelle_balkenanzahl, min_balkenanzahl, max_balkenanzahl: ℕ
anzeigeeinstellung, log_datei, pop_up_menu: STATUS

Zähler

Das Schema *Zaehler* stellt im Modell den Detektor dar, der den aktuell gemessenen Wert der Photonen beinhaltet. Die erste Idee bei der Spezifikation war den Zähler nicht in einem eigenen Schema darzustellen, sondern zusammen mit dem Zählerfenster. Es musste die Frage beantwortet werden, ob der Zähler ohne aktives Zählerfenster aktiv ist. Die akustische Ausgabe arbeitet nicht bei Testzählern und konnte somit nicht zur Lösung beitragen. Bei aktiviertem Zählerfenster kann der Nutzer das Aufzeichnen der Zählraten auswählen (Log-Datei). Schließt man das Zählerfenster, endet jedoch auch die Datenaufzeichnung. Demnach findet sich auch hier keine Antwort auf die soeben gestellte Frage. Eine eindeutige Formulierung ist in der verbalen Spezifikation [5] gegeben:

„ 2.2 Start des Zählvorganges

...Der Start dieser Programmfunktion kann auf zwei unterschiedliche Arten erfolgen:

1. *automatisch nach dem Programmstart, wenn der Parameter "Startup" im [Steuerprogramm]-Abschnitt der xcontrol.ini-Datei entsprechend gesetzt ist, oder*
2. *über Menüfunktionen: Entweder über das Hauptmenü Einstellungen > Detektoren > Detektoren ... oder über das PopUp-Menü im Zählerfenster Einstellungen > Gerät ...*

Zu 1.: Ist der Startup-Parameter entsprechend gesetzt (s. Pkt. 3.2.1.a), wird nach der Programm-Initialisierungsphase das Zählerfenster automatisch

geöffnet und es werden die Zählraten des ersten verfügbaren Detektors angezeigt. ...“

Der zuerst beschriebene Fall kann bestätigt werden. Das Zählerfenster wird geöffnet und Zählraten werden angezeigt.

„Zu 2.: Start über das Hauptmenü Einstellungen > Detektoren > Detektoren ... oder über das PopUp-Menü im Zählerfenster Einstellungen > Gerät ....

*Fall 2a***:** *Erstmaliger Aufruf der Menüfunktion und Startup-Parameter war nicht entsprechend gesetzt*
Ist der Startup-Parameter nicht entsprechend gesetzt und damit auch kein Zählvorgang nach dem Programmstart gestartet worden, erfolgt der Start des Zählvorganges bei der erstmaligen Wahl der Hauptmenüfunktion: Einstellungen > Detektoren > Detektoren ...
Gestartet wird der erste verfügbare Detektor, der in der Dialogbox auch angezeigt wird ... Das Zählerfenster wird nicht automatisch geöffnet, sondern erst nach dem Setzen des Markierungs-Feldes Fenster > Anzeigen...“

Aus Fall 2a kann man entnehmen, dass mit dem Ausführen der Hauptmenüfunktion: „Einstellungen > Detektoren > Detektoren ...“ ein Zählvorgang gestartet wird, obwohl dieser nicht visuell dargestellt wird. Aufgrund dieser Beschreibung habe ich mich dazu entschlossen den Zähler separat zu modellieren.

Zaehler

$XCTL$
$Zeit$
$aktueller\_wert: \mathbb{N}$
$beginn\_zaehlung : HUNDERTSTEL$

$t - beginn\_zaehlung \leqslant aktuelle\_zeitbegrenzung$
$aktueller\_wert \leqslant aktuelle\_impulsbegrenzung$

Im Deklarationsteil des Schemas befindet sich eine Variable t vom Typ *HUNDERTSTEL*. Sie steht im Zusammenhang mit der Modellierung der Zeit. *HUNDERTSTEL* ist ein Basistyp, mit dem Hundertstelsekunden modelliert werden sollen. ‚t’ gibt die Zeit an, die seit dem Programmstart von XCTL vergangen ist. Die Variable ‚*beginn_zaehlung*’ ist ebenfalls von Typ

HUNDERTSTEL. Mit ihr wird der Startzeitpunkt eines Zählvorganges festgehalten. Die Beschreibung des verwendeten Zeitmodells erfolgt in Abschnitt 2.3.3.

Initialzustand

Die Modellierung des Initialzustandes soll nachfolgend näher betrachtet werden. Hervorzuheben ist, dass die Anzahl der Detektoren im System nicht festgelegt ist. Die Überprüfung, ob und wie viele Detektoren vorhanden sind, erfolgt in der Initialisierungsphase von XCTL. Eine Modellierung wurde im Schema *Init_XCTL* vorgenommen.

Init_XCTL

In der Initialisierungsdatei muss laut [5] der erste Detektor die Kennzahl 0 aufweisen, die folgenden müssen aufsteigend nummeriert sein (device_x, mit x = 0,1,2,…). Ist kein Detektor 0 vorhanden bricht das Programm ab.
Im Modell wird ein Typ Detektor definiert: *DETEKTOR* ::= *detektor* ⟪ℕ⟫.
Die Elemente sind alle Detektoren mit einer natürlichen Zahl als Kennzahl. Jeder Detektor der vom System erkannt wird, wird im Modell der Menge „*Aktive Detektoren'* " zugeordnet.
Für jeden Detektor sind eine Zeitbegrenzung, eine Impulsbegrenzung und der Status für die akustische Wiedergabe in der Initialisierungsdatei gesichert. Diese Werte werden als geordnete Paare dargestellt. Beim Start des Programms ist standardmäßig der Detektor 0 ausgewählt, entsprechend werden die Zeit- und Intervallbegrenzung sowie der Soundparameter gesetzt.
Die Beschränkung für die Einbeziehung von Detektoren in das System lautet:

$$\forall x: \mathbb{N} \mid$$
$$\bullet\ \textit{detektor}\ x \in \textit{Aktive\_Detektoren'}$$
$$\Leftrightarrow x = 0 \vee (\exists y: \mathbb{N} \mid \textit{detektor}\ y \in \textit{Aktive\_Detektoren'} \bullet x - y = 1)$$

Bedeutung: Für alle natürlichen Zahlen x soll gelten:
Der Detektor mit der Kennzahl x ist genau dann Element der Menge *Aktive_Detektoren'* wenn gilt: x = 0. Oder eine natürliche Zahl y existiert, die Kennzahl eines Detektors ist, der wiederum Element der Menge *Aktive_Detektoren'* ist, so dass gilt: Die Differenz von x und y ist eins.

Ein Detektor wird also nur aufgenommen, wenn sein „Vorgänger", dessen Kennzahl um eins kleiner ist als die eigene, schon erkannt wurde. Das setzt

ebenfalls voraus, dass ein Detektor mit der Kennzahl null existiert und erkannt wurde.

Init_XCTL

$$
\begin{array}{l}
INI\_File \\
XCTL' \\
Anzeige\_Zaehlerfenster' \\
Zaehler' \\
\Delta Zeit \\
\hline
\forall x: \mathbb{N} \\
\quad \bullet\ detektor\ x \in Aktive\_Detektoren' \\
\quad\ \Leftrightarrow x = 0 \lor (\exists y: \mathbb{N} \mid detektor\ y \in Aktive\_Detektoren' \bullet x - y = 1) \\
Aktive\_Detektoren' \neq \emptyset \\
zaehlerfenster' = Aus \Leftrightarrow xcontrol\_startup = Startup\_Nothing \\
zaehlerfenster' = Ein \land Init\_Anzeige\_Zaehlerfenster \land Init\_Zaehler \\
\Leftrightarrow \neg\ xcontrol\_startup = Startup\_Nothing \\
konfigurationsfenster' = Aus \\
aktueller\_detektor' = detektor\ 0 \\
\exists z: DETEKTOR \times HUNDERTSTEL \mid \\
\quad z \in Zeitbegrenzungen\_Default \land z\ .\ 1 = detektor\ 0 \\
\quad\ \bullet\ aktuelle\_zeitbegrenzung' = z\ .\ 2 \\
\exists i: DETEKTOR \times \mathbb{N} \mid i \in Impulsbegrenzungen\_Default \land i\ .\ 1 = detektor\ 0 \\
\quad \bullet\ aktuelle\_impulsbegrenzung' = i\ .\ 2 \\
\exists s: DETEKTOR \times STATUS \mid s \in Soundeinstellung\_Default \\
\qquad\qquad \land s\ .\ 1 = detektor\ 0 \bullet sound' = s\ .\ 2 \\
t' = 0
\end{array}
$$

Zusätzlich zu den schon erwähnten Beschränkungen wird aufgrund eines entsprechend gesetzten „Startup“-Parameters (*xcontrol_startup)* das Zählerfenster entweder geöffnet oder nicht und der Zählvorgang gestartet.
Das Schema Zeit wird ebenfalls geändert. Die Zeitvariable t’ soll im Folgezustand den Wert 0 besitzen, womit ein Startzeitpunkt für das XCTL-Programm definiert wird.

Init_Anzeige_Zählerfenster

Das Schema zeigt, welche Werte die Variablen besitzen, wenn das Zählerfenster erstmalig aktiviert wird.

Init_Anzeige_Zaehlerfenster

Anzeige_Zaehlerfenster'

---

aktuelle_intensitaet' = 140001

default_intensitaet' = 260000

min_intensitaet' = *min_balkenanzahl'* = 0

max_intensitaet' = 2000000

max_balkenanzahl' = 140

log_datei' = *Aus*

Init_Zaehler

Dieses Schema beschreibt den Initialzustand des Zählers.

Init_Zaehler

Zaehler'

Ξ*Zeit*

---

beginn_zaehlung' = *t*

aktueller_wert' = 0

Zu Beginn eines Zählvorgangs ist der aktuelle Wert der gezählten Photonen Null. Die Zählung beginnt zum Zeitpunkt t.

Nach den Zustandsbeschreibungen erfolgt im nächsten Schritt die Identifikation von Operationen.

Operationen

Insgesamt konnten 18 Operationen festgehalten werden, die in verschiedenen Schemata modelliert wurden. Dazu gehören:

- das Einblenden/Ausblenden des Konfigurationsfensters

- das Aktivieren/Deaktivieren der Speicherung von Messdaten
- das Ändern der Anzeige
- das Einblenden/Ausblenden und Ändern der Anzeigeeinstellungen
- die Änderung der Soundoption
- die Änderung der aktuellen Zeitbegrenzung
- die Änderung der aktuellen Impulsbegrenzung
- das Einblenden/Ausblenden des Zählerfensters
- die Detektorauswahl
- der Zählvorgang
- der Zählerneustart (zeit- und impulsbedingt)
- das Beenden von XCTL

Einige dieser Operationschemata sollen im Folgenden gezeigt werden. Die vollständige formale Spezifikation ist in Anhang A.2 zu finden.

Aktivieren/Deaktivieren von Messdaten Speichern

Das Aktivieren bzw. Deaktivieren der Aufzeichnung von Messdaten soll als erstes Beispiel für die Modellierung herangezogen werden. In der Grundeinstellung ist die Option nicht aktiviert, so dass keine Messdaten gespeichert werden. Tests haben ergeben, dass das Speichern mit Schließen des Zählerfensters deaktiviert wird. Daher erfolgte die Deklaration der Statusvariable *log_datei* im Schema *Anzeige_Zaehlerfenster*.

Das Operationsschema *Messdaten_Speichern_Aktivieren* ist wie folgt definiert:

Messdaten_Speichern_Aktivieren

Δ*Anzeige_Zaehlerfenster*
ereignis?: EREIGNISSE

---

zaehlerfenster = Ein
konfigurationsfenster = Aus
pop_up_menu = Ein
pop_up_menu' = Aus
ereignis? = Markierung_Log_Datei
log_datei = Aus
log_datei' = Ein

Eine Reihe von Bedingungen muss erfüllt sein, in erster Linie in Bezug auf den Status dargestellter Fenster. Es handelt sich somit um ein recht einfaches Operationsschema. Für das Deaktivieren dieser Option existiert ebenfalls ein eigenes Schema mit einem Unterschied in den letzten beiden Zeilen des Prädikatteils.

Messdaten_Speichern_Deaktivieren

Δ*Anzeige_Zaehlerfenster*
ereignis?: EREIGNISSE

---

zaehlerfenster = Ein
konfigurationsfenster = Aus
pop_up_menu = Ein
pop_up_menu' = Aus
ereignis? = Markierung_Log_Datei
log_datei = Ein
log_datei' = Aus

Die Modellierung hätte ebenfalls in nur einem Schema dargestellt werden können, indem die letzten beiden Zeilen des Prädikatteils der Schemata kombiniert werden. (verkürzt dargestellt)

Messdaten_Option_Ändern

Δ*Anzeige_Zaehlerfenster*
ereignis?: EREIGNISSE

…..
log_datei = Ein $\wedge$ *log_datei' = Aus* $\vee$ *log_datei =Aus* $\wedge$ *log_datei' =Ein*

Da es sich bei der Option um eine Art Schalter handelt, ist dieser entweder in der Position ‚Ein' oder ‚Aus' und nach dem Betätigen dementsprechend ‚Aus' oder ‚Ein'.

Zählerneustart (zeitbedingt)

Die letzten Modellierungen von Operationen, die vorgestellt werden sollen, sind der zeitbedingte und der impulsbedingte Neustart des Zählers. Der Zählvorgang läuft, sofern gestartet, bis eine bestimmte Zeit erreicht ist oder eine bestimmte Impulszahl gemessen wurde.

Das Schema *Zaehler_Neustart_Zeitbedingt* beschreibt den Fall des zeitbedingten Neustarts:

Zaehler_Neustart_Zeitbedingt

$\Delta Zaehler$

$\Xi Zeit$

$output!: \mathbb{N}$

$z: DETEKTOR \times \mathbb{N}$

$t - beginn\_zaehlung > aktuelle\_zeitbegrenzung$

$aktueller\_wert' = 0$

$beginn\_zaehlung' = t$

$output! = aktueller\_wert$

$aktuelle\_zeitbegrenzung' = sicherung\_zeitbegrenzung$

Der gezählte Wert der Impulse wird ausgegeben (*output! = aktueller_wert*). Zusätzlich wird die Zeitbegrenzung neu eingelesen, da ein impulsbedingter Neustart des Zählvorgangs eine Auswirkung auf die aktuelle Zeitbegrenzung hat bzw. gehabt haben könnte.

Der impulsbedingte Neustart des Zählers wird wie folgt beschrieben:

Zaehler_Neustart_Impulsbedingt

$\Delta Zaehler$

$\Xi Zeit$

$output!: \mathbb{N}$

$aktueller\_wert > aktuelle\_impulsbegrenzung$

$$
\begin{array}{|l}
output! = aktueller\_wert \\
aktuelle\_zeitbegrenzung' = aktuelle\_zeitbegrenzung - (t - beginn\_zaehlung) \\
\wedge\ beginn\_zaehlung' = t \\
aktueller\_wert' = 0 \\
\hline
\end{array}
$$

Wird der Zähler aufgrund der Impulsbegrenzung neu gestartet, verringert sich die aktuelle Zeitbegrenzung um die Zeit, die bisher abgelaufen ist.
Durch die visuelle Ausgabe kann man diesen Fakt nicht gut nachvollziehen, daher möchte ich hier auf [4] hinweisen:

2.3.1 b
"Eine technisch bedingte maximale Zählrate des Detektors ist ein zweites Abbruchkriterium für den Detektor….Dadurch kann die tatsächliche Messzeit geringer ausfallen (weil bereits der Impulsbegrenzungswert erreicht ist), als die vorgegebene Zeit-Begrenzung. Um trotzdem die Zählrate für das vorgegebene Messzeit-Intervall anzeigen zu können, zählt der Detektor in einem weiteren Intervall bei Null beginnend so lange, bis die Summe aus der Länge des ersten Intervalls plus der des zweiten Intervalls der vorgegebenen Zeitbegrenzung entspricht. Reicht auch dieses noch nicht aus, beginnt ein drittes Intervall usw."

Diese Beschreibung war die Grundlage der Modellierung des impulsbedingten Neustarts des Zählers.

2.3 Der Zeitaspekt

Die Z-Notation ist eine unter vielen existierenden Spezifikationstechniken. Ihr Einsatz bei der Spezifikation großer kommerzieller Systeme hat sich bewährt, wie zum Beispiel im CICS-Projekt (Custumer Information Control System) der Firma IBM in den 80er Jahren[9] und bei Mondex, einem elektronischen Zahlungssystem für Chipkarten[10], bei dem die Sicherheitseigenschaften mit Hilfe von Z bewiesen werden konnten, wodurch das Systems mit dem höchsten Sicherheitslevel der ITSEC[11] (E6) klassifiziert werden konnte.
Eine Schwäche, die der Notation vorgehalten wird, sind fehlende explizite Funktionen, um zeitliche Beschränkungen zu beschreiben. (siehe [6], [7], [8])

[9] http://en.wikipedia.org/wiki/CICS
[10] http://en.wikipedia.org/wiki/Mondex
[11] Information Technology Security Evaluation Criteria der Komission der EU

2.3.1 Das Intervall Modell

Andrew Coombes und John McDermid haben in ihrer Arbeit: „*Specifying Temporal Requirements for Distributed Real-Time Systems in Z*" [8] Definitionen vorgestellt, die es ermöglichen sollen die zeitlichen Eigenschaften mit Z zu beschreiben. Die in dieser Arbeit erstellte Spezifikation von Manuelle Justage (Alt) nutzt den grundlegenden Teil dieses Ansatzes: Das Intervallmodell.

In [8] wird eine vollständige formale Beschreibung des Modells gegeben. Einige Relationen und Funktionen, die definiert wurden, sollen hier betrachtet werden.

Der Basistyp des Modells ist die Menge aller Intervalle.

[INTERVAL]

Die Ordnungsrelation 'before' wird dargestellt durch:

syntax *before inrel before*

| _before_: INTERVAL ↔ INTERVAL
|——————————————
| ∀a, b, c: INTERVAL • a before b ∧ b before c ⇒ a before c
| ∀a: INTERVAL • ¬ a before a
| ∀a, b: INTERVAL • ¬ (a before b ∧ b before a)
| ∃a, b: INTERVAL | a ≠ b • ¬ a before b ∧ ¬ b before a

Die Zeile „**syntax** *before inrel before*" ist nicht Bestandteil der originalen Formalisierung. Der Editor[12], der zur Erstellung der Spezifikationen genutzt wurde, verlangt jedoch, dass die Bezeichner von definierten Relationen und Funktionen in obiger Form bekannt gemacht werden müssen. Das Wort ‚*inrel*' beschreibt dabei den Typ der Relation - eine Infix-Relation. Details zur Funktionsweise und der Arbeit mit dem Editor sind in dessen Nutzerhandbuch verfügbar (siehe [13]). Die Definition der Relation ‚before' erfolgt in einer ‚axiomatic box', einem Strukturmittel der Z-Notation. Ebenso wie ein Schema besteht diese aus einem Deklarationsteil und einem Prädikatteil. Die

[12] Z/Eves – Die Internetseite des Herstellers existiert nicht mehr.

Beschränkungen im Prädikatteil bestimmen die Eigenschaften der Relation. Es handelt sich um eine transitive, irreflexive, asymmetrische Relation, also eine strikte Halbordnung. Mit dieser Relation kann beschrieben werden, dass beispielsweise ein Intervall A vor einem Intervall B vorkommt.

Die Relation ‚*during*' ist ebenfalls eine Ordnungsrelation und wie folgt definiert:

syntax *during inrel during*

$$
\begin{array}{|l}
\_during\_ : INTERVAL \leftrightarrow INTERVAL \\
\hline
\forall b, c: INTERVAL \\
\quad \bullet\ b\ during\ c \\
\qquad \Leftrightarrow (\forall a, d: INTERVAL \mid a\ before\ c \land c\ before\ d \bullet a\ before\ b \\
\qquad\qquad\qquad\qquad\qquad\qquad \land\ b\ before\ d)
\end{array}
$$

Mit ihr soll eine Aussage darüber getroffen werden können, ob ein Intervall X während eines Intervalls Y vorkommt.

Eine weitere Ordnungsrelation, die das Überlappen von zwei Intervallen beschreibt, ist mit der folgenden Beschreibung gegeben:

syntax *overlaps inrel overlaps*

$$
\begin{array}{|l}
\_overlaps\_ : INTERVAL \leftrightarrow INTERVAL \\
\hline
\forall a, b: INTERVAL \bullet a\ overlaps\ b \\
\qquad\qquad \Leftrightarrow (\exists c: INTERVAL \bullet c\ during\ a \land c\ during\ b)
\end{array}
$$

(Anmerkung: Der Name '*overlaps*' wurde von mir gewählt, da das ursprüngliche Symbol für die Funktion '//' vom Z-Editor nicht als Name erkannt wurde.)

Die Funktion ‚*period*' ist wie folgt definiert:

$$
\begin{array}{|l}
period: INTERVAL \rightarrow \mathbb{N} \\
\hline
\forall i: INTERVAL \\
\quad \bullet \ ((point\ i) \Rightarrow period\ i = 0) \\
\qquad \wedge\ (\neg\ (point\ i) \Rightarrow period\ i = \#\ (PerfectClock\ During\ i))
\end{array}
$$

‚*period*' wird definiert als eindeutige Abbildung von der Menge der Intervalle auf die Menge der natürlichen Zahlen. Die im Prädikatteil vorkommende Funktion ‚*point i*' liefert als Ergebnis ein minimales Intervall. Das ist ein Intervall, das nicht mehr geteilt werden kann. ‚*PerfectClock*' ist die im Modell definierte Uhr. Sie ist definiert als eine Menge von Intervallen, deren Elemente sich nicht überlappen dürfen. Das bedeutet, für je zwei beliebige Intervalle A und B dieser Menge muss gelten, dass das Intervall A ‚aufhört' bevor B ‚beginnt' und umgekehrt. Zusätzlich wird verlangt, dass nur minimale Intervalle als Element der Menge ‚*PerfectClock*' erlaubt sind. Weiterhin erscheint eine Funktion ‚During'. Diese liefert als Ergebnis eine Menge von Intervallen, die gleichzeitig zu einem Intervall i existieren.

Was besagt nun die Funktion ‚period'?

Einem gegebenen Intervall i wird eine natürliche Zahl zugeordnet.

Fall 1: Ist i ein minimales Intervall,
so gilt laut Definition: period i = 0.

Fall 2: i ist ein Intervall, jedoch nicht minimal, so ist period i = x, wobei x eine natürliche Zahl ist, die der Anzahl der Elemente entspricht, die die Funktion ‚During' zurückgibt. Also die Anzahl der minimalen Intervalle, die gleichzeitig mit dem Intervall i vorkommen.

Wenn man sich vorstellt, dass ein minimales Intervall dem ‚Tick' einer Uhr entspricht, so wird einem beliebigen nicht minimalen Intervall die Anzahl der ‚Ticks' zugeordnet, die vorkommen, solange das Intervall andauert. Weißt man dem Zeitraum zwischen zwei ‚Ticks' eine Zeiteinheit zu (z.B. eine Sekunde), liefert die Funktion ‚period' die Länge eines Intervalls in Sekunden.

2.3.2 Der Zeitaspekt in „Manuelle Justage (Alt)“

In der Spezifikation von der Funktion ‚Manuelle Justage (Alt)’ erscheint die Funktion ‚period’ in den Operationsschemata, die den Fahrbetrieb beschreiben. Die Länge (zeitlich) der Eingabe des Nutzers ist von Interesse. Die Darstellung erfolgt für den Antrieb DC.

Fahrbetrieb_DC

Δ*Manuelle_Justage*
eingabe?: EREIGNISSE
laenge_der_eingabe: INTERVAL

---

manuelle_justage = Ein
aktueller_antrieb = DC
aktuelle_antriebsart_dc = fahrbetrieb
if *eingabe? = Links* $\wedge$ *dc_geschwindigkeit_aktuell* > 0
$\wedge$ *dc_rel_ist_position*
*- dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
$<$ *dc_rel_position_min*
then *dc_rel_ist_position' = dc_rel_position_min*
else *dc_rel_ist_position'*
$=$ *dc_rel_ist_position*
*- dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
$\vee$ (**if** *eingabe? = Recht* $\wedge$ *dc_geschwindigkeit_aktuell* > 0
$\wedge$ *dc_rel_ist_position*
*+ dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
$>$ *dc_rel_position_max*
then *dc_rel_ist_position' = dc_rel_position_max*
else *dc_rel_ist_position'*
$=$ *dc_rel_ist_position*
*+ dc_geschwindigkeit_aktuell * period laenge_der_eingabe*)
$\vee$ *dc_geschwindigkeit_aktuell* $= 0$
$\wedge$ *eingabe? = Links*
$\wedge$ *dc_rel_ist_position' = dc_rel_position_min*
$\vee$ *dc_geschwindigkeit_aktuell* $= 0$
$\wedge$ *eingabe? = Rechts*

$\wedge$ *dc_rel_ist_position' = dc_rel_position_max*

Erfolgt eine Eingabe ‚Links' und sind die drei ersten Bedingungen erfüllt und die aktuelle Geschwindigkeit ist größer Null, so ergibt sich die neue Position aus der alten Position, abzüglich des Produktes der aktuellen Geschwindigkeit und der Länge der Eingabe. Die Länge der Eingabe wird durch *period laenge_der_eingabe* beschrieben. Dabei ist die Variable *laenge_der_eingabe* vom Typ *INTERVAL* und stellt den Zeitraum dar, in dem der Nutzer die Eingabe tätigte. Eine weitere Voraussetzung für eine Bewegung während des gesamten Intervalls ist, dass die untere Bewegungsgrenze nicht überschritten wird. Für die Eingabe ‚Rechts' wird dies analog dargestellt.

Eine weitere Einbindung der Zeit in die Beschreibung von „Manuelle Justage (Alt)" ist noch nicht erfolgt. Das Intervallmodell wurde verwendet, da es die Länge eines gegebenen Intervalls bestimmen kann. Im Modell ist das die Länge der Eingabe. Eine weitere Modellierungsidee wäre: Die Zeit als Intervall darzustellen in der sich ein Antrieb in Bewegung befindet. Die Länge des Intervalls kann durch die aktuelle Geschwindigkeit und die Länge des Weges von der aktuellen Position zur Zielposition bestimmt werden.
Dadurch wäre es möglich, eine gleichzeitige Bewegung der Antriebe DF und DC beispielsweise mit der Bedingung zu beschränken, dass sich Bewegungsintervalle der beiden Antriebe nicht überlappen dürfen.
Zudem könnte die Eingabe von Parametern eines Antriebes während eines bestimmten Zeitraumes verboten werden. Das wäre das Intervall, während dessen sich dieser Antrieb in Bewegung befindet.

2.3.3 Der Zeitaspekt in „0-dimensionale Detektoren, Zählerfenster"

In der Spezifikation der Teilfunktion „0-dimensionale Detektoren, Zählerfenster" findet man einen Zeitaspekt in den Schemata *‚Zaehler'*, *‚Init_Zaehler'*, *‚Zaehler_Neustart_Zeitbedingt'* und *‚Zaehler_Neustart_Impulsbedingt'*.

Das Zeitmodell, welches hier zugrunde liegt, unterscheidet sich von dem vorangegangenem. In der Veröffentlichung: „Z and the Specification of Real-Time Systems" von Bruel, Benzekri und Raymond [6] wird es verwendet, um einen periodischen Datenstrom zu spezifizieren.

Das Zeitmodell aus [6]

Die Zeit wird modelliert mit dem Schema *Zeit* und dem Operationsschema *Tick.*

Um den Zeitaspekt im Modell zu verdeutlichen, werden Zeitangaben durch den Basistyp *HUNDERTSTEL* beschrieben, welcher der Menge der natürlichen Zahlen entspricht.

$$HUNDERTSTEL == \mathbb{N}$$

Zeit

$$t : HUNDERTSTEL$$

Tick

$$\Delta Zeit$$

$$t' = t + 1$$

Das Fortschreiten der Zeit soll durch das Schema *Tick* dargestellt werden. Die gegenwärtige Zeit wird durch die Variable t beschrieben. Das Erhöhen der Variablen t geschieht im Modell unendlich oft. Die Zeit kann außer von der Operation *Tick* nicht beeinflusst werden, da keine andere Operation den Wert der Variable t verändert.

Die Zeit in der Spezifikation „0-dimensionale Detektoren, Zählerfenster“

Im Initialzustand des Systems hat t den Wert Null. Das wird durch die Bedingung t’ = 0 im Schema *Init_XCTL* modelliert.

Im Schema *Init_Zaehler* wird der Beginn eines Zählvorgangs gespeichert.

Init_Zaehler

$$Zaehler'$$

$$\Xi Zeit$$

$$beginn\_zaehlung' = t$$

$$aktueller\_wert' = 0$$

Zaehler

$XCTL$

$Zeit$

$aktueller\_wert: \mathbb{N}$

$beginn\_zaehlung, ende\_zaehlung: HUNDERTSTEL$

$t - beginn\_zaehlung \leqslant aktuelle\_zeitbegrenzung$

$aktueller\_wert \leqslant aktuelle\_impulsbegrenzung$

Eine Beschränkung im Schema *Zaehler* besagt, dass die Differenz von ‚t' und ‚beginn_zaehlung' kleiner gleich der aktuellen Zeitbegrenzung sein muss. Wird diese Bedingung nicht erfüllt, wird die aktuell gezählte Impulsrate ausgegeben.

Dieser Fall wird dargestellt im Schema *Zaehler_Neustart_Zeitbedingt*:

Zaehler_Neustart_Zeitbedingt

$\Delta Zaehler$

$\Xi Zeit$

$output!: \mathbb{N}$

$z: DETEKTOR \times \mathbb{N}$

$t - beginn\_zaehlung > aktuelle\_zeitbegrenzung$

$aktueller\_wert' = 0$

$beginn\_zaehlung' = t$

$output! = aktueller\_wert$

$aktuelle\_zeitbegrenzung' = sicherung\_zeitbegrenzung$

Das hier verwendete Zeitmodell ist relativ einfach, genügt jedoch um den Zeitaspekt von [5] zu beschreiben. Der Vorteil dieses Modells ist, dass eine Spezifikation durch Einbringung des Zeitaspektes nicht wesentlich komplexer wird. Die Lesbarkeit bleibt erhalten. Das in Abschnitt 2.3.1 vorgestellte Intervallmodell erhöht hingegen die Komplexität der Spezifikation.

3. Probleme der verbalen Spezifikationen

Im Rahmen der formalen Verhaltensspezifikation der Teilssysteme „Manuelle Justage (Alt)“ und „0-dimensionale Detektoren, Teilfunktion Zählerfenster“ wurden die verbalen Beschreibungen intensiv gelesen und kritisch betrachtet. Im Ergebnis konnten in [4] diverse Ungenauigkeiten entdeckt werden, die im Folgenden detailliert dargestellt werden sollen. In [5] wurde demgegenüber nur ein Kritikpunkt bezüglich der Beschreibung des Systemverhaltens entdeckt.
Zusätzlich möchte ich auf Beobachtungen des Programmverhaltens hinweisen, die während der Untersuchung der Software festgestellt wurden und noch nicht von [4] und [5] erfasst wurden. Damit konnte festgestellt werden, dass diese Dokumente an verschieden Stellen unvollständig waren.

3.1 Manuelle Justage (Alt)

Anhand von Zitaten aus der verbalen Verhaltensspezifikation soll auf identifizierte Probleme hingewiesen werden. Ein Problem der Bewertung von verbalen Beschreibungen ist das subjektive Verständnis, ob es sich bei einer bestimmten Formulierung um eine Ungenauigkeit, eine Mehrdeutigkeit oder sogar um einen Fehler handelt.

1. Beispiel: Punkt 2.1 Antriebe zur Positionierung der Probe

Auszug aus dem Pflichtenheft:

> „*...Alle Werte in der Dialogbox beziehen sich dann auf diesen aktuellen Antrieb, wodurch zu einem Zeitpunkt immer nur ein Antrieb durch die Dialogbox gesteuert werden kann.*“

Problem: Mehrdeutigkeit

Was bedeuteten in diesem Zusammenhang „gesteuert“ und „Zeitpunkt“? Eine Interpretation wäre: Zu einem „Zeitpunkt“ können für einen Antrieb die spezifischen Werte editiert und der Antrieb in Bewegung gesetzt werden. Muss das Ende der Bewegung abgewartet werden? Oder kann man während der Bewegung die Werte eines anderen Antriebes ändern und ihn in Bewegung setzen?

Das Verhalten des Systems wird in Punkt 2.2 der verbalen Verhaltensspezifikation eindeutig geklärt. Hier sollte gezeigt werden, wie die Worte „gesteuert“ und „Zeitpunkt“ interpretiert werden könnten.

2. Beispiel: Punkt 2.3c Betriebsart und Start der Bewegung

Auszug aus dem Pflichtenheft:

„*Für jeden Antrieb gibt es drei Betriebsarten, um die gewünschte Soll-Position anzusteuern:*
...
2. Schrittbetrieb: ... Die Bewegung wird aktiviert, wenn eine der Cursortasten der Tastatur (<- oder ->) betätigt wird bzw. auf dem Scrollbar das linke oder rechte Endeelement angeklickt wird. ...
3. Fahrbetrieb: Der Antrieb wird solange bewegt, wie die Cursortasten (<- oder ->) der Tastatur oder die Endeelemente des Scrollbars gedrückt gehalten werden. ...“

Problem: Ungenauigkeit

Für beide Betriebsarten wird der Antrieb aktiviert, wenn die linke oder rechte Pfeiltaste oder das linke oder rechte Endelement der Laufleiste (Scrollbar) betätigt werden. Es fehlt der Hinweis, dass die jeweilige Antriebsart aktiviert werden muss, falls dies nicht so ist bevor die Bewegung gestartet wird.

→ Beobachtung

Wird bei aktiviertem Schrittbetrieb eine Richtungstaste bzw. das entsprechende Endelement der Laufleiste dauerhaft betätigt, erfolgen mehrere aufeinander folgende Schritte. Zum Verhältnis zwischen Dauer und Anzahl der Schritte kann keine genaue Aussage getroffen werden.

3. Beispiel: Punkt 2.3d Bewegungsgeschwindigkeit

Auszug aus dem Pflichtenheft:

„*In den Betriebsarten Direktbetrieb und Fahrbetrieb kann die Bewegungsgeschwindigkeit des Antriebs vorgegeben werden. Sie kann vom Benutzer im Eingabefeld Fahren mit V = ... s-1 eingetragen werden. Im Schrittbetrieb wird der Antrieb immer mit der maximal möglichen Geschwindigkeit bewegt....*“

Problem: Fehler

Hier wurde ein Fehler in der Verhaltensbeschreibung entdeckt. Die Geschwindigkeit, mit der sich ein Antrieb im Schrittbetrieb bewegt, ist unter bestimmten Vorraussetzungen nicht maximal.

Fall 1: Schrittbetrieb aktiviert

Wird eine neue zulässige Bewegungsgeschwindigkeit eingegeben, beispielsweise mit der Absicht per Direktbetrieb eine bestimmte Position zu erreichen, so gilt diese ebenfalls für den Schrittbetrieb. Wird nach der Eingabe oder nach einer Bewegung im Direktbetrieb der Schrittbetrieb aktiviert, bewegt sich der Antrieb mit der eingegebenen Geschwindigkeit.
Fehler**:** Die Geschwindigkeit im Schrittbetrieb sollte maximal sein!
Erfolgt ein Wechsel in den Fahrbetrieb, bleibt die ausgewählte Geschwindigkeit bestehen (korrekt). Erfolgt ein Wechsel zurück in den Schrittbetrieb, wird der Geschwindigkeitswert auf sein Maximum gesetzt (korrekt für Schrittbetrieb). Dies wirkt sich jedoch auf den Direktbetrieb aus. Hier würde eine Bewegung nach dem beschriebenen Wechsel der Antriebsarten mit maximaler Geschwindigkeit durchgeführt werden. Für den Direktbetrieb wäre dies kein Fehler, da die Geschwindigkeit für den Nutzer sichtbar ist. Gleichwohl wäre es angebracht ihn darüber zu informieren

Fall 2: Fahrbetrieb aktiviert

Wird eine neue zulässige Bewegungsgeschwindigkeit eingegeben, so gilt diese für den Fahrbetrieb und den Direktbetrieb. Erfolgt ein Wechsel in den Schrittbetrieb, wird die Geschwindigkeit auf das Maximum gesetzt. Dies hat auch Auswirkungen auf den Direktbetrieb (siehe Fall 1). Erfolgt ein Wechsel zurück in den Fahrbetrieb, wird die Geschwindigkeit auf den zuvor eingegebenen Wert gesetzt.
Eine Lösungsmöglichkeit wäre, dass der im Eingabefeld für die Geschwindigkeit dargestellte Wert keinen Einfluss auf den Schrittbetrieb hat. Das würde auch die Änderung der aktuellen Geschwindigkeit beim Wechsel der Antriebsarten überflüssig machen. Zumindest sollte der Nutzer über eine Änderung informiert werden.

→ **Beobachtung: Bewegungsgeschwindigkeit gleich Null**

Erfolgt in dem Geschwindigkeitsfeld die Eingabe Null, wird ein nicht erwartetes Verhalten des Systems beobachtet.

Fall 1: Direktbetrieb wird aktiviert

Nachdem die Eingabe einer neuen Position in das Feld ‚Neuer Winkel' erfolgt ist und der Direktbetrieb mit ‚Enter' aktiviert wird, geschieht optisch nichts. Bis dahin verhält sich das System wie man es erwartet. Wird allerdings jetzt eine Einstellung vorgenommen, beispielsweise die Antriebsart gewechselt, erfolgt ein ‚Sprung' zur eingegebenen Position.

Fall 2: Fahrbetrieb wird aktiviert

Wird der Fahrbetrieb durch Betätigung einer Richtungstaste aktiviert, geschieht wie in Fall 1 optisch nichts. Wird allerdings eine Einstellung vorgenommen, erfolgt auch hier ein ‚Sprung'. Je nach dem, ob die linke oder rechte Bewegung ausgelöst wurde, befindet sich der Antrieb nunmehr auf der minimalen oder maximalen Position.

Fall 3: Schrittbetrieb

Ein ähnliches Verhalten ist zu beobachten, wenn der Schrittbetrieb aktiviert wird. Die „Sprungweite" entspricht der aktuellen Schrittweite, während die Richtung von der Eingabe, die den Schrittbetrieb aktiviert hat, anhängig ist. Die Veränderung wird auch hier erst nach einer weiteren Einstellung sichtbar.

Um dieses Problem zu umgehen, könnte man das Geschwindigkeitsminimum größer Null festlegen. Ohnehin ist eine Bewegung mit einer Geschwindigkeit gleich Null nicht sinnvoll.

4. Beispiel: Punkt 2.3e Reaktion auf unzulässige Eingaben, 3. unzulässige Schrittweite

Auszug aus dem Pflichtenheft:

„*…Falls durch die Nutzung einer zulässigen Schrittweite der Bereich für zulässige Winkelpositionen verlassen wird, muss ebenfalls eine Korrektur der Schrittweite vorgenommen werden.*"

Problem: Ungenauigkeit

Aus diesem Satz geht nicht hervor wie sich das System verhält.

→ **Beobachtung**

Wird oder ist eine zulässige Schrittweite eingegeben mit der durch einen Schritt die zulässige Winkelposition verlassen werden würde, so erfolgt bei der Aktivierung des Schrittbetriebes keine Reaktion (entsprechende Bewegungsrichtung vorausgesetzt).

5. Beispiel: Punkt 3.3 Zulässige Motorpositionen (absolute Positionen)

Auszug aus dem Pflichtenheft:

„*…Außerdem gibt es zwei Konstanten im ini-File für die Auflösungsweite: AngleWidth, PositionWidth. …*“

Problem: Ungenauigkeit

Zu Beginn einer Sitzung entspricht die hier benannte Konstante „AngleWidth“ dem in der Dialogbox angezeigten Wert für die Schrittweite.

In diesem Unterpunkt wird auf die Rundung der eingegebenen Werte hingewiesen. Diese Rundungen sind auch mit dem Verhaltenstest der Software nicht eindeutig nachvollziehbar.
Im Ini-File existieren Koeffizienten, die jeweils einen Umrechnungsfaktor darstellen. Wie diese Rundung erfolgt, ist nicht ersichtlich.

6. Beispiel: Punkt 3.5 Bewegungsgeschwindigkeit

→ **Beobachtung**

Die eingegebenen Werte für die Geschwindigkeit werden beim Wechsel der Antriebsarten gerundet.

Zusammenfassung:

Im Dokument wurden fünf Positionen gefunden, in denen größere Probleme mit der verbalen Spezifikation aufgetreten sind. In Beispiel 6 wird nur eine Beobachtung wiedergegeben.

3.2 0-dimensionale Detektoren, Teilfunktion Zählerfenster

Aus den Erfahrungen der „Manuellen Justage (Alt)“ hätte erwartet werden können, dass auch hier eine Reihe von Ungenauigkeiten entdeckt würde. Das Verhalten des Systems ist jedoch korrekt beschrieben. Der Test der Software wurde ohne angeschlossene Hardware durchgeführt. Einen Punkt möchte ich dennoch diskutieren: Das Systemverhalten in Bezug auf den Zählvorgang, speziell auf den erstmaligen Start und das Ende.

Die verbale Beschreibung [5] enthält dazu folgende Informationen:

„ *2.2 Start des Zählvorganges*

Start des Zählvorganges heißt das Auslesen der Zählraten aus einem verfügbaren Detektor. Der Start dieser Programmfunktion kann auf zwei unterschiedliche Arten erfolgen:

1. *automatisch nach dem Programmstart, wenn der Parameter "Startup" im [Steuerprogramm]-Abschnitt der xcontrol.ini-Datei entsprechend gesetzt ist, oder*
2. *über Menüfunktionen: Entweder über das Hauptmenü Einstellungen > Detektoren > Detektoren ... oder über das PopUp-Menü im Zählerfenster Einstellungen > Gerät ...*

***Zu 1.:** Ist der Startup-Parameter entsprechend gesetzt (s. Pkt. 3.2.1.a), wird nach der Programm-Initialisierungsphase das Zählerfenster automatisch geöffnet und es werden die Zählraten des ersten verfügbaren Detektors ...“*

Der erste hier aufgeführte Fall kann eindeutig nachvollzogen werden, da die Zählrate optisch angezeigt wird.

„***Zu 2.:** Start über das Hauptmenü Einstellungen > Detektoren > Detektoren ... oder über das PopUp-Menü im Zählerfenster Einstellungen > Gerät ....*

***Fall 2a:** Erstmaliger Aufruf der Menüfunktion und Startup-Parameter war nicht entsprechend gesetzt*
Ist der Startup-Parameter nicht entsprechend gesetzt und damit auch kein Zählvorgang nach dem Programmstart gestartet worden, erfolgt der Start des Zählvorganges bei der erstmaligen Wahl der Hauptmenüfunktion: Einstellungen > Detektoren > Detektoren ...
Gestartet wird der erste verfügbare Detektor, der in der Dialogbox auch angezeigt wird (s. Pkt. 2.3.1.a). Ist zusätzlich der Sound-Parameter in dem

entsprechenden Device-Abschnitt gesetzt, erfolgt außerdem eine von den Zählraten abhängige akustische Ausgabe (s. Pkt. 2.3.1.d). Das Zählerfenster wird nicht automatisch geöffnet, sondern erst nach dem Setzen des Markierungs-Feldes Fenster > Anzeigen (s. Pkt. 2.3.1.c).“

Mit der Auswahl von „Einstellungen > Detektoren > Detektoren“ aus dem Hauptmenü in Fall 2a öffnet sich das Zählerkonfigurationsfenster. Nach dem Wortlaut der Beschreibung ist der Zählvorgang jetzt gestartet worden. Bestätigt werden kann diese Beschreibung nicht. Das Zählerfenster wird nicht angezeigt, daher kann keine optische Verifikation erfolgen. Wenn der Soundparameter gesetzt ist (Sound: ein) kann keine Beurteilung vorgenommen werden, da nur simulierte Zähler zur Verfügung stehen (keine akustische Ausgabe möglich). Ist der Soundparameter nicht gesetzt (Sound: aus), stellt sich die Frage: Woher weiß man, ob der Zählvorgang aktiviert ist?
Eine Rettung könnte die Funktion ‚log_datei’ sein, welche die aktuellen Messwerte in einer Datei speichert. Man wird jedoch auch hier enttäuscht, da das Speichern der Messwerte nur mit aktiviertem Zählerfenster möglich ist. (Vgl. 2.7 in [5])

Fall 2b: *Kein erstmaliger Aufruf einer Menüfunktion bzw. der Startup-Parameter war entsprechend gesetzt.*
In diesem Fall ist bereits ein Zählvorgang zu dem in der Dialogbox angezeigten Detektor gestartet. Bei der Auswahl eines anderen oder des gleichen Detektors wird der bisherige Detektor beendet und damit auch das Auslesen der Zählraten, und der neue Detektor wird gestartet.

Der Fall 2b bezieht sich nicht auf das erstmalige Starten eines Zählvorganges.

4. Auswertung

Der Nutzen von formalen Spezifikationen bei der Entwicklung von Software ist aus qualitativer Sicht unbestritten. Die Wirtschaftlichkeit ihrer Anwendung stellt jedoch oft ein Problem dar. Die formale Spezifikation und insbesondere der Beweis von Systemeigenschaften bedeuten meist hohen Aufwand und demzufolge höhere Kosten. Würden Beweise aus Aufwands- oder Kostengründen weggelassen, wird es sicherlich viele Stimmen geben, die den ganzen Prozess in Frage stellen. Vorteile kann die Nutzung formaler Methoden trotzdem bringen. Wie diese Arbeit zeigt, konnten Unklarheiten in den nicht formalen Beschreibungen identifiziert werden.

Die Spezifikationen wurden mit der Software Z/Eves erzeugt. Diese bietet neben einem Z-Editor die Möglichkeit, das Modell sowohl syntaktisch als auch semantisch zu überprüfen. Eine Beweisfunktion wird zwar ebenfalls bereitgestellt, diese dient dem Nutzer jedoch nur zur Unterstützung. Der Editor und die syntaktische Überprüfung der Eingabe erwiesen sich als sehr hilfreich. Die Homepage des Herstellers der Software ist nicht mehr erreichbar bzw. zeigt andere Inhalte an.

Der Aufwand des Erstellens der Spezifikationen in diesem Reverse-Engineering-Prozess muss als hoch eingestuft werden. Viele Informationen, die für die formale Umsetzung von Bedeutung waren, konnten aus den verbalen Beschreibungen entnommen werden: Wie viele Antriebe existieren? Welche Intervallgrenzen liegen vor? Oder welche Parameter existieren? Jedoch mussten umfangreiche Tests mit der Software durchgeführt werden. Zum einen, um unklare Beschreibungen zu verstehen und zum anderen, um Informationen zu verifizieren.

Ein zweiter Gesichtspunkt betrifft die Kenntnisse über formale Methoden. Je mehr Routine im Umgang mit einer Sprache oder Notation besteht, desto schneller erfolgt die Umsetzung einer Problemstellung. Für die Z-Notation sind vielfältige Informationen und Werkzeuge verfügbar, und die Nutzung eines Editors für diese Notation ist von immensem Vorteil. Der entstandene Aufwand ist jedoch höher als erwartet ausgefallen.

Die Einbeziehung des Zeitaspektes mit dem Intervallmodell erwies sich ebenfalls als sehr komplex, was daran liegen mag, dass die Z-Notation nicht die optimale Methode zur Beschreibung des Zeitaspektes ist. Die Idee war, wie schon in 2.3.2 angedeutet, mit Hilfe der Intervalle gleichzeitige Bewegungsabläufe zu beschreiben. Das einfachere Zeitmodell, das in der formalen Spezifikation von [5] eingesetzt wird, würde dazu nicht genügen. Die Umsetzung ist hier noch nicht gelungen und wurde daher nicht einbezogen.

Daraus könnte man folgende Schlussfolgerung ziehen: Umso detaillierter die formale Beschreibung des Verhaltens eines Systems stattfindet, umso komplexer wird die formale Spezifikation und umso höher ist der Aufwand.
Eine formale Beschreibung sollte zum besseren Verständnis informale Kommentare enthalten, die dann ebenfalls in der verbalen Beschreibung genutzt werden können. Um die jeweiligen Vorteile zu nutzen, könnten beide Dokumente sogar in Kombination erstellt werden. In Reverse-Engineering-Prozessen sollen Dokumente erstellt werden, um ein vorhandenes System zu „verstehen“. Zum Verständnis des Verhaltens der untersuchten Systeme ist denkbar, dass eine Kombination beider Beschreibungstechniken vorteilhaft ist. Das grobe Verhalten eines Systems lässt sich mit der Umgangssprache leicht und verständlich darstellen, so dass der Leser einen ersten Eindruck erhält. Wird die Beschreibung detailreicher, steigt der Nutzen der formalen Beschreibungstechnik. Das formale Modell kann sicherlich logische Zusammenhänge besser verdeutlichen.
Wichtig ist die Verbindung beider Dokumente. In der verbalen Beschreibung sollten direkte Verweise auf die formale Spezifikation erfolgen und umgekehrt sollten in der formalen Beschreibung Kommentare nicht fehlen, die in Bezug zur verbalen Beschreibung stehen.
Hat sich der Einsatz der Z-Notation in diesem Reverse-Engineering-Prozess bewährt?
Aufgrund der festgestellten Fehler in [4] kann die Frage eindeutig bejaht werden. In [5] konnte dagegen nur ein Punkt „diskutiert“, jedoch keine Fehler bezüglich des Systemverhaltens aufgedeckt werden.

5. Ausblick

Dass sich die Spezifikationen positiv auf das Projekt auswirken, und folglich auch auf die Qualität der anschließenden Arbeiten, kann nur gehofft werden. Der Einsatz formaler Methoden im Rahmen des XCTL-Projektes könnte weiter fortgeführt werden. Ein mögliches Ziel ist die formale Spezifikation des gesamten XCTL-Systems. Dazu sollte eine formale Beschreibung aller Teilssysteme erfolgen, die im späteren Verlauf zu einer XCTL-Spezifikation zusammengefasst werden können.
Weiterhin könnte man die bisher entstandenen formalen Spezifikationen einer kritischen Betrachtung unterziehen. Mit Hilfe von Programmtests müsste dabei überprüft werden, ob das System korrekt spezifiziert wurde. Die Einbeziehung des Zeitaspektes hinsichtlich der „Manuellen Justage“, speziell bei der gleichzeitigen Bewegung von Antrieben, wäre ebenfalls denkbar.
Es existieren demnach noch viele Möglichkeiten, formale Spezifikations-methoden in diesem Reverse-Engineering-Prozess einzusetzen.

Anhang

A.1 Formale Spezifikation „Manuelle Justage (Alt)“

Hinweise zur Spezifikation

Die Spezifikation wurde kommentiert. Kommentare sind am Beginn durch „//**“ und am Ende durch „**//“ gekennzeichnet. Die Spezifikation wurde mit dem Editor der Software Z/Eves, in der Programmversion 2.1 erstellt. Bei der Übertragung des „Intervallmodells“ [8] mussten leichte syntaktische Veränderungen vorgenommen werden.

*//**
Abschnitt1 – Intervall-Modell
Der folgende Abschnitt wurde vollständig übernommen, um den Zeitaspekt in das Modell aufzunehmen.
Einige wenige Details der Syntax wurden an das Werkzeug angepasst.
***//*

[*INTERVAL*]

syntax *before inrel before*

$$\_before\_ : INTERVAL \leftrightarrow INTERVAL$$

$$\forall a, b, c: INTERVAL \bullet a\ before\ b \land b\ before\ c \Rightarrow a\ before\ c$$
$$\forall a: INTERVAL \bullet \neg\ a\ before\ a$$
$$\forall a, b: INTERVAL \bullet \neg\ (a\ before\ b \land b\ before\ a)$$
$$\exists a, b: INTERVAL \mid a \neq b \bullet \neg\ a\ before\ b \land \neg\ b\ before\ a$$

theorem axiom *BEGIN*
$\exists begin: INTERVAL \bullet \forall b: INTERVAL \bullet \neg\ b\ before\ begin$

theorem axiom *NOEND*
$\forall a: INTERVAL \bullet \exists b: INTERVAL \bullet a\ before\ b$

theorem axiom *DENSE*
$\forall a, b: INTERVAL \mid a\ before\ b \bullet \exists c: INTERVAL \bullet a\ before\ c \land c\ before\ b$

syntax *during inrel during*

| *_during_: INTERVAL ↔ INTERVAL*
|---
| *∀b, c: INTERVAL*
| • *b during c*
| ⇔ (*∀a, d: INTERVAL* | *a before c ∧ c before d* • *a before b ∧ b before d*)

theorem axiom *ATOM*
∀a: INTERNAL • *∃b: INTERNAL* | *b during a* • *∀c: INTERNAL* | *c during b* • *c = b*

theorem axiom *MOLECULAR*
∃a, b: INTERVAL | *b ≠ a* • *b during a*

syntax *overlaps inrel overlaps*

| *_overlaps_: INTERVAL ↔ INTERVAL*
|---
| *∀a, b: INTERVAL* • *a overlaps b* ⇔ (*∃c: INTERVAL* • *c during a ∧ c during b*)

theorem axiom *FINITE*
∀i: INTERVAL; I: ℙ INTERVAL | *¬* (*∃j, k: I* • (*j ≠ k ∧ j overlaps k*))
• *∃n: ℕ* • *n* = # { *j: I* | *j during i* }

theorem axiom *NOCONT*
∃a: INTERVAL • *∀b, c: INTERVAL* | *b during a ∧ c during a* • *b overlaps c*

theorem axiom *LINEARITY*
∀a, b: INTERVAL • *a before b ∨ b before a ∨ a overlaps b*

syntax *point prerel point*

$$
\begin{array}{|l}
point\_: \mathbb{P}\ INTERVAL \\
\hline
\forall i: INTERVAL \bullet (point\ i) \Leftrightarrow \neg (\exists j: INTERVAL \mid i \neq j \bullet j\ during\ i)
\end{array}
$$

$$
\begin{array}{|l}
start, end: INTERVAL \rightarrow INTERVAL \\
\hline
\forall i: INTERVAL \\
\quad \bullet \exists j: INTERVAL \mid j\ during\ i \wedge (point\ j) \\
\qquad \bullet (start\ i = j \Leftrightarrow \neg (\exists k: INTERVAL \mid k\ during\ i \wedge k \neq j \bullet k\ before\ j)) \\
\qquad \wedge (end\ i = j \Leftrightarrow \neg (\exists k: INTERVAL \mid k\ during\ i \wedge k \neq j \bullet j\ before\ k))
\end{array}
$$

syntax *compose infun2 compose*

$$
\begin{array}{|l}
\_compose\_: INTERVAL \times INTERVAL \rightarrow INTERVAL \\
\hline
\forall i1, i2: INTERVAL \\
\quad \bullet \exists i3: INTERVAL \mid i1\ during\ i3 \wedge i2\ during\ i3 \\
\qquad \bullet i1\ compose\ i2 = i3 \\
\qquad \Leftrightarrow \neg (\exists j: INTERVAL \mid i1\ during\ j \wedge i2\ during\ j \\
\qquad\qquad \bullet (j \neq i3 \wedge j\ during\ i3))
\end{array}
$$

syntax *intersect infun2 intersect*

$$
\begin{array}{l}
\quad \_intersect\_ : INTERVAL \times INTERVAL \nrightarrow INTERVAL \\
\hline
\quad \forall i1, i2 : INTERVAL \mid i1 \ overlaps \ i2 \\
\quad\quad \bullet \ \exists i3 : INTERVAL \mid i3 \ during \ i1 \wedge i3 \ during \ i2 \\
\quad\quad\quad \bullet \ i1 \ intersect \ i2 = i3 \\
\quad\quad\quad\quad \Leftrightarrow \neg \ (\exists j : INTERVAL \mid j \ during \ i1 \wedge j \ during \ i2 \\
\quad\quad\quad\quad\quad\quad \bullet \ (j \neq i3 \wedge i3 \ during \ j))
\end{array}
$$

$$
\begin{array}{l}
INTSET \ == \\
\quad \{ \ is : \mathbb{P} \ INTERVAL \\
\quad\quad \mid \forall i, j : INTERVAL \mid i \in is \wedge j \in is \wedge j \neq i \bullet \neg \ i \ overlaps \ j \ \}
\end{array}
$$

$$
\begin{array}{l}
\quad begin : INTSET \rightarrow INTERVAL \\
\hline
\quad \forall I : INTSET \\
\quad\quad \bullet \ \exists_1 i : INTERVAL \mid i \in I \\
\quad\quad\quad \bullet \ begin \ I = i \Leftrightarrow \neg \ (\exists j : INTERVAL \mid j \in I \bullet j \ before \ i)
\end{array}
$$

syntax *While infun4 While*

syntax *During infun4 During*

syntax *Later infun4 Later*

syntax *Earlier infun4 Earlier*

$$
\begin{array}{l}
\quad \_Earlier\_ : INTSET \times INTERVAL \rightarrow INTSET \\
\quad \_Later\_ : INTSET \times INTERVAL \rightarrow INTSET \\
\quad \_During\_ : INTSET \times INTERVAL \rightarrow INTSET \\
\quad \_While\_ : INTERVAL \times INTSET \rightarrow INTSET \\
\hline
\quad \forall i : INTERVAL; I : INTSET \\
\quad\quad \bullet \ I \ Earlier \ i = \{ \ j : INTERVAL \mid j \in I \wedge j \ before \ i \ \} \\
\quad\quad\quad \wedge \ I \ Later \ i = \{ \ j : INTERVAL \mid j \in I \wedge i \ before \ j \ \} \\
\quad\quad\quad \wedge \ I \ During \ i = \{ \ j : INTERVAL \mid j \in I \wedge j \ during \ i \ \} \\
\quad\quad\quad \wedge \ i \ While \ I = \{ \ j : INTERVAL \mid j \in I \wedge i \ during \ j \ \} \\
\quad\quad\quad \wedge \ \# \ (i \ While \ I) \leqslant 1
\end{array}
$$

syntax *rd infun5 rd*

syntax *th infun5 th*

syntax *st infun5 st*

syntax *nd infun5 nd*

$$
\begin{array}{l}
\quad \_th\_,\ \_st\_,\ \_nd\_,\ \_rd\_ : \mathbb{N} \times INTSET \rightarrow \mathbb{N} \times INTSET \\
\hline
\quad \forall n : \mathbb{N};\ i : INTSET \\
\quad\quad \bullet\ n\ st\ i = (n, i) \wedge n\ nd\ i = (n, i) \wedge n\ rd\ i = (n, i) \wedge n\ th\ i = (n, i)
\end{array}
$$

syntax *After infun3 After*

syntax *Before infun3 Before*

$$
\begin{array}{l}
\quad \_After\_ : (\mathbb{N} \times INTSET) \times INTERVAL \nrightarrow INTERVAL \\
\quad \_Before\_ : (\mathbb{N} \times INTSET) \times INTERVAL \nrightarrow INTERVAL \\
\hline
\quad \forall I : INTSET;\ i : INTERVAL;\ n : \mathbb{N} \mid I \in \mathbb{F}\ INTERVAL \\
\quad\quad \bullet\ (\#\{\ j : INTERVAL \mid j \in I \wedge i\ before\ j\ \} \geqslant n \\
\quad\quad\quad \Rightarrow n\ th\ I\ After\ i \\
\quad\quad\quad\quad = (\mu j : INTERVAL \mid j \in I \wedge \#\,(I\ During\ (j\ compose\ i)) = n \bullet j)) \\
\quad\quad\quad \wedge (\#\{\ j : INTERVAL \mid j \in I \wedge j\ before\ i\ \} \geqslant n \\
\quad\quad\quad\quad \Leftrightarrow n\ th\ I\ Before\ i \\
\quad\quad\quad\quad\quad = (\mu j : INTERVAL \mid j \in I \wedge \#\,(I\ During\ (j\ compose\ i)) = n \bullet j))
\end{array}
$$

syntax *nS postfun nS*

syntax *muS postfun muS*

syntax *mS postfun mS*

syntax *S postfun S*

$$
\begin{array}{l}
\quad PerfectClock: INTSET \\
\quad \_nS: \mathbb{N} \rightarrow \mathbb{N} \\
\quad \_muS: \mathbb{N} \rightarrow \mathbb{N} \\
\quad \_mS: \mathbb{N} \rightarrow \mathbb{N} \\
\quad \_S: \mathbb{N} \rightarrow \mathbb{N} \\
\hline
\quad \forall i: INTERVAL \mid i \in PerfectClock \bullet point\ i \\
\quad \forall x: \mathbb{N} \bullet (x * 1000)\ nS = x\ muS \land (x * 1000)\ muS = x\ mS \\
\qquad \land \quad (x * 1000)\ mS = x\ S
\end{array}
$$

$$
\begin{array}{l}
\quad period: INTERVAL \rightarrow \mathbb{N} \\
\hline
\quad \forall i: INTERVAL \\
\qquad \bullet ((point\ i) \Rightarrow period\ i = 0) \\
\qquad\quad \land (\neg (point\ i) \Rightarrow period\ i = \#(PerfectClock\ During\ i))
\end{array}
$$

*//***
Abschnitt 2: Die Spezifikation von Manuelle Justage
***//*

*//** Eingaben die vom Nutzer getätigt werden. Als freier Typ definiert. **//*

EREIGNISSE ::= *Links*
| *Enter*
| *Rechts*
| *Ausfuehren_Manuelle_Justage_Alt*
| *Verlassen_Manuelle_Justage*
| *Verlassen_XCTL*
| *Setze_Relative_Null*
| *Relative_Null_Aufheben*

*//** wählbaren Betriebsmöglichkeiten **//*

ANTRIEBSART ::= *fahrbetrieb* | *schrittbetrieb*

*//** die einzelnen Antriebe / Motoren **//*

ANTRIEB ::= DC | DF | AR | TL | CC

*//** Statustyp **//*

STATUS ::= Ein | Aus

*//**
INI_File*

Deklaration der Variablen, die aus dem ini-File (hardware.ini) gelesen werden (nur relevante für Manuelle Justage (Alt)).

dc_ist_position ≙ aktuelle Position des Motors DC (Beugung Grob) (Bezeichner in der Ini-Datei: DeltaPosition)
dc_schrittweite ≙ aktuelle Schrittweite des Motors DC (Bezeichner in der Ini-Datei: AngleWidth)
dc_position_min ≙ minimale Winkelposition des Motors DC (Bezeichner in der Ini-Datei: AngleMin)
dc_position_max ≙ maximale Winkelposition des Motors DC (Bezeichner in der Ini-Datei: AngleMax)
dc_schrittweite_min ≙ minimale Schrittweite des Motors DC (Bezeichner in der Ini-Datei: MinimalWidth)
dc_schrittweite_max ≙ maximale Schrittweite des Motors DC (Bezeichner in der Ini-Datei: MaximalWidth)
dc_geschwindigkeit ≙ aktuelle Geschwindigkeit des Motors DC (Bezeichner in der Ini-Datei: Velocity)
dc_geschwindigkeit_max ≙ maximale Geschwindigkeit des Motors DC (Bezeichner in der Ini-Datei: MaxVelocity)

Die Bezeichnung der Variablen für die anderen Antriebe erfolgt analog.
***//*

INI_File

```
dc_ist_position, dc_position_min, dc_position_max, dc_schrittweite,
dc_schrittweite_min, dc_schrittweite_max, dc_geschwindigkeit,
dc_geschwindigkeit_max: ℤ
df_ist_position, df_position_min, df_position_max, df_schrittweite,
df_schrittweite_min, df_schrittweite_max, df_geschwindigkeit,
df_geschwindigkeit_max: ℤ
ar_ist_position, ar_position_min, ar_position_max, ar_schrittweite,
ar_schrittweite_min, ar_schrittweite_max, ar_geschwindigkeit,
ar_geschwindigkeit_max: ℤ
tl_ist_position, tl_position_min, tl_position_max, tl_schrittweite,
tl_schrittweite_min, tl_schrittweite_max, tl_geschwindigkeit,
tl_geschwindigkeit_max: ℤ
cc_ist_position, cc_position_min, cc_position_max, cc_schrittweite,
cc_schrittweite_min, cc_schrittweite_max, cc_geschwindigkeit,
cc_geschwindigkeit_max: ℤ
```

*//***
XCTL - Deklaration von Variablen, die verfügbar sind solange das XCTL - Steuerprogramm läuft.

dc_rel_null ≙ relative Null des Motors DC (initial ist relative Null gleich absoluter Null)
dc_rel_ist_position ≙ relative ist_Position (relativ zur relativen Null)
dc_rel_position_min, dc_rel_position_max ≙ minimale und maximale relative Winkelposition

analog für die anderen Antriebe
***//*

XCTL

INI_File

dc_rel_null, dc_rel_ist_position, dc_rel_position_min, dc_rel_position_max,
df_rel_null, df_rel_ist_position, df_rel_position_min, df_rel_position_max,
ar_rel_null, ar_rel_ist_position, ar_rel_position_min, ar_rel_position_max,
tl_rel_null, tl_rel_ist_position, tl_rel_position_min, tl_rel_position_max,
cc_rel_null, cc_rel_ist_position, cc_rel_position_min,
cc_rel_position_max: ℤ
aktueller_antrieb: ANTRIEB
manuelle_justage: STATUS

---

dc_rel_position_min = dc_position_min - dc_rel_null
dc_rel_position_max = dc_position_max - dc_rel_null
df_rel_position_min = df_position_min - df_rel_null
df_rel_position_max = df_position_max - df_rel_null
ar_rel_position_min = ar_position_min - ar_rel_null
ar_rel_position_max = ar_position_max - ar_rel_null
tl_rel_position_min = tl_position_min - tl_rel_null
tl_rel_position_max = tl_position_max - tl_rel_null
cc_rel_position_min = cc_position_min - cc_rel_null
cc_rel_position_max = cc_position_max - cc_rel_null

*//**
Manuelle Justage

Deklaration von Variablen, die während "Manuelle Justage (Alt)" aktiviert ist, verfügbar sind.
dc_schrittweite_aktuell ≙ aktuelle Schrittweite des Motors DC
dc_geschwindigkeit_aktuell ≙ aktuelle Geschwindigkeit des Motors DC
dc_geschwindigkeit_aktuell_2 ≙ eine zweite aktuelle Geschwindigkeit des Motors DC,
nötig da beim Wechsel zwischen Schrittbetrieb und Fahrbetrieb die aktuelle Geschwindigkeit variiert
aktuelle_antriebsart_dc ≙ aktuell gewählte Antriebsart des Motors DC
andere Motoren analog
aktueller_antrieb ≙ der aktuell gewählte Motor
**//

```
Manuelle_Justage
XCTL
dc_schrittweite_aktuell, dc_geschwindigkeit_aktuell,
dc_geschwindigkeit_aktuell_2, df_schrittweite_aktuell,
df_geschwindigkeit_aktuell, df_geschwindigkeit_aktuell_2,
ar_schrittweite_aktuell, ar_geschwindigkeit_aktuell,
ar_geschwindigkeit_aktuell_2, tl_schrittweite_aktuell,
tl_geschwindigkeit_aktuell, tl_geschwindigkeit_aktuell_2,
cc_schrittweite_aktuell, cc_geschwindigkeit_aktuell,
cc_geschwindigkeit_aktuell_2: ℤ
aktuelle_antriebsart_dc, aktuelle_antriebsart_df, aktuelle_antriebsart_ar,
aktuelle_antriebsart_tl, aktuelle_antriebsart_cc: ANTRIEBSART
```

*//**
Initialisierung von XCTL

für alle Motoren gilt:
relative Null gleich absoluter Null
relative aktuelle Position gleich der aktuellen Position (aus dem Ini-File)
aktueller Motor ist DF (abhängig von der Position der Beschreibung im Ini-File)
***//*

Init_XCTL

INI_File
XCTL'

dc_rel_null' = 0
df_rel_null' = 0
ar_rel_null' = 0
tl_rel_null' = 0
cc_rel_null' = 0
dc_rel_ist_position' = dc_ist_position
df_rel_ist_position' = df_ist_position
ar_rel_ist_position' = ar_ist_position
tl_rel_ist_position' = tl_ist_position
cc_rel_ist_position' = cc_ist_position
aktueller_antrieb' = DF
manuelle_justage' = Aus

*//**
Initialisierung Manuelle Justage

Aktuelle Schrittweite, aktuelle Geschwindigkeit sowie die Art des Antriebs werden festgelegt.
***//*

Init_Manuelle_Justage

XCTL

Manuelle_Justage'

---

dc_schrittweite_aktuell' = dc_schrittweite

dc_geschwindigkeit_aktuell' = dc_geschwindigkeit_max

dc_geschwindigkeit_aktuell_2' = dc_geschwindigkeit_max

aktuelle_antriebsart_dc' = schrittbetrieb

df_schrittweite_aktuell' = df_schrittweite

df_geschwindigkeit_aktuell' = df_geschwindigkeit_max

df_geschwindigkeit_aktuell_2' = df_geschwindigkeit_max

aktuelle_antriebsart_df' = schrittbetrieb

ar_schrittweite_aktuell' = ar_schrittweite

ar_geschwindigkeit_aktuell' = ar_geschwindigkeit_max

ar_geschwindigkeit_aktuell_2' = ar_geschwindigkeit_max

aktuelle_antriebsart_ar' = schrittbetrieb

tl_schrittweite_aktuell' = tl_schrittweite

tl_geschwindigkeit_aktuell' = tl_geschwindigkeit_max

tl_geschwindigkeit_aktuell_2' = tl_geschwindigkeit_max

aktuelle_antriebsart_tl' = schrittbetrieb

cc_schrittweite_aktuell' = cc_schrittweite

cc_geschwindigkeit_aktuell' = cc_geschwindigkeit_max

cc_geschwindigkeit_aktuell_2' = cc_geschwindigkeit_max

aktuelle_antriebsart_cc' = schrittbetrieb

---

*//***

Manuelle Justage (Alt) - Starten

***//*

Start_Manuelle_Justage

ΔXCTL

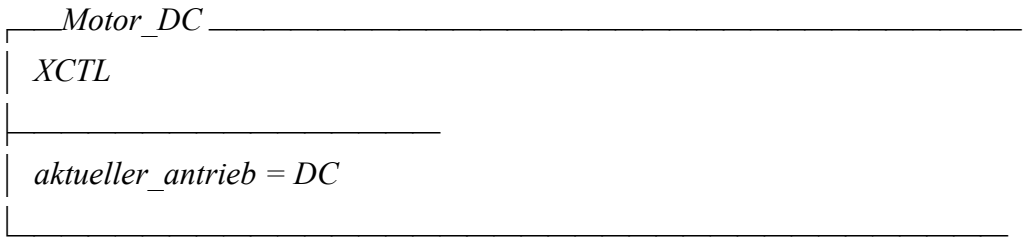
ΔManuelle_Justage

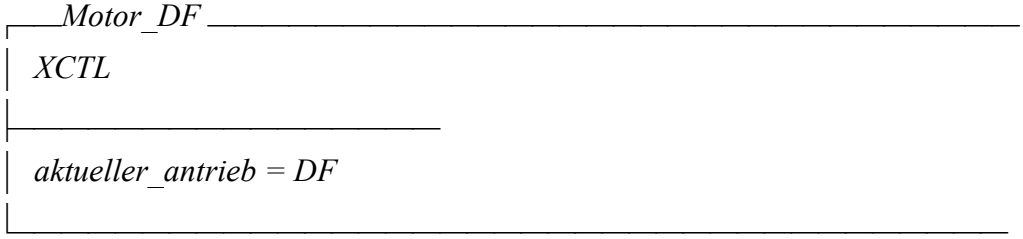
∃ereignis?: EREIGNISSE | ereignis? = Ausfuehren_Manuelle_Justage_Alt

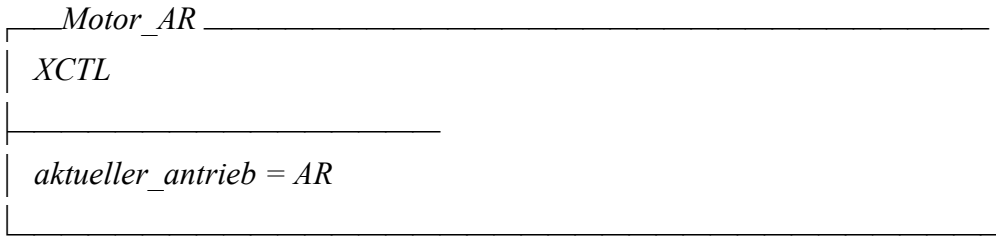
- *Init_Manuelle_Justage ∧ manuelle_justage = Ein*

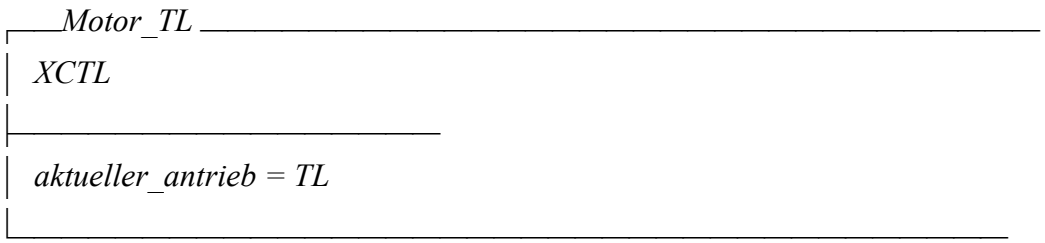
---

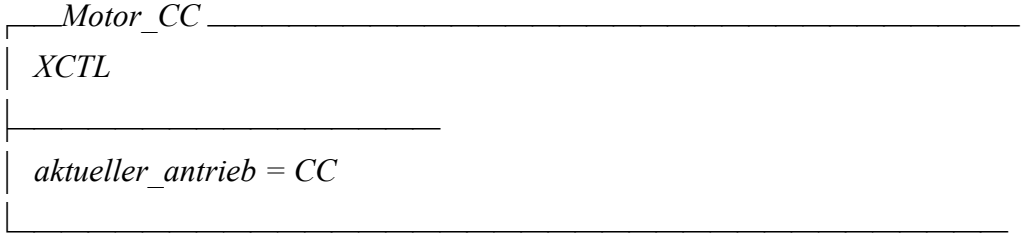
*//** Motor_..*
Strukturmittel, zur späteren Kombination von Schemata zu vollständigen Operationen.
jeweils für einen Motor
Verwendung noch fraglich ... noch zu klären
***//*











*//***
Auswahl Antrieb

Ein Antrieb wird ausgewählt.
***//*

Auswahl_Antrieb

ΔXCTL
neuer_antrieb?: ANTRIEB

---

manuelle_justage = Ein
aktueller_antrieb' = neuer_antrieb?

*//***
Auswahl_Antriebsart

Eine neue Antriebsart wird gewählt, dabei erfolgt eine Änderung der aktuellen Geschwindigkeit.
am Beispiel von Motor DC:
Die neue Antriebsart entspricht im Folgezustand der Eingabe.
Befindet man sich im aktuellen Zustand im Fahrbetrieb und soll ein Wechsel zum Schrittbetrieb vorgenommen werden, so ändert sich die aktuelle Geschwindigkeit (wird auf Maximum gesetzt).
Erfolgt ein Wechsel vom Schrittbetrieb zum Fahrbetrieb, so wird die aktuelle Geschwindigkeit auf die Geschwindigkeit des Fahrbetriebes gesetzt.

analog für die anderen Motoren
***//*

Auswahl_Antriebsart_DC

Δ*Manuelle_Justage*

neue_antriebsart?: ANTRIEBSART

---

manuelle_justage = Ein

aktueller_antrieb = DC

aktuelle_antriebsart_dc' = neue_antriebsart?

if *aktuelle_antriebsart_dc = fahrbetrieb* $\wedge$ *neue_antriebsart? = schrittbetrieb*

then *dc_geschwindigkeit_aktuell' = dc_geschwindigkeit_max*

if *aktuelle_antriebsart_dc = schrittbetrieb*

$\wedge$ *neue_antriebsart? = fahrbetrieb*

then *dc_geschwindigkeit_aktuell' = dc_geschwindigkeit_aktuell_2*

*//***
analog für die restlichen Motoren
***//*

Auswahl_Antriebsart_DF $\widehat{=}$

Auswahl_Antriebsart_DC[*aktuelle_antriebsart_df'*/*aktuelle_antriebsart_dc'*,
aktuelle_antriebsart_df/*aktuelle_antriebsart_dc*,
df_geschwindigkeit_aktuell'/*dc_geschwindigkeit_aktuell'*,
df_geschwindigkeit_aktuell_2/*dc_geschwindigkeit_aktuell_2*,
df_geschwindigkeit_max/*dc_geschwindigkeit_max*]

Auswahl_Antriebsart_AR $\widehat{=}$

Auswahl_Antriebsart_DC[*aktuelle_antriebsart_dar'*/*aktuelle_antriebsart_dc'*,
aktuelle_antriebsart_ar/*aktuelle_antriebsart_dc*,
ar_geschwindigkeit_aktuell'/*dc_geschwindigkeit_aktuell'*,
ar_geschwindigkeit_aktuell_2/*dc_geschwindigkeit_aktuell_2*,
ar_geschwindigkeit_max/*dc_geschwindigkeit_max*]

Auswahl_Antriebsart_TL $\widehat{=}$

Auswahl_Antriebsart_DC[*aktuelle_antriebsart_tl'*/*aktuelle_antriebsart_dc'*,
aktuelle_antriebsart_tl/*aktuelle_antriebsart_dc*,
tl_geschwindigkeit_aktuell'/*dc_geschwindigkeit_aktuell'*,
tl_geschwindigkeit_aktuell_2/*dc_geschwindigkeit_aktuell_2*,
tl_geschwindigkeit_max/*dc_geschwindigkeit_max*]

Auswahl_Antriebsart_CC ≙

Auswahl_Antriebsart_DC[*aktuelle_antriebsart_cc'*/*aktuelle_antriebsart_dc'*,
aktuelle_antriebsart_cc/*aktuelle_antriebsart_dc*,
cc_geschwindigkeit_aktuell'/*dc_geschwindigkeit_aktuell'*,
cc_geschwindigkeit_aktuell_2/*dc_geschwindigkeit_aktuell_2*,
cc_geschwindigkeit_max/*dc_geschwindigkeit_max*]

*//***
Direktbetrieb

Wird ein neuer Winkel eingegeben und diese Eingabe mit "Enter" bestätigt, so wird
der Direktbetrieb des aktuellen Motors aktiviert.
Die Geschwindigkeit muss größer Null sein (sonst irreguläres Verhalten - noch zu klären).
Der Motor bewegt sich zur eingegebenen Position. Liegt der eingegebene Winkel außerhalb der Grenzen, so wird der Motor bis zur minimalen bzw. maximalen Position bewegt.
***//*

Direktbetrieb_DC

Δ*Manuelle_Justage*
neuer_winkel?: ℤ
eingabe?: *EREIGNISSE*
—
manuelle_justage = *Ein*
aktueller_antrieb = *DC*
eingabe? = *Enter*
if *neuer_winkel?* < *dc_rel_position_min*
then *dc_rel_ist_position'* = *dc_rel_position_min*
else if *neuer_winkel?* > *dc_rel_position_max*
then *dc_rel_ist_position'* = *dc_rel_position_max*
else *dc_rel_ist_position'* = *neuer_winkel?*

Direktbetrieb_DF ≙

Direktbetrieb_DC[*df_geschwindigkeit_aktuell*/*dc_geschwindigkeit_aktuell*,
df_rel_position_min/*dc_rel_position_min*,
df_rel_position_max/*dc_rel_position_max*,
df_rel_istposition/*dc_rel_ist_position*]

Direktbetrieb_AR ≙

Direktbetrieb_DC[*ar_geschwindigkeit_aktuell*/*dc_geschwindigkeit_aktuell*,
ar_rel_position_min/*dc_rel_position_min*,
ar_rel_position_max/*dc_rel_position_max*,
ar_rel_istposition/*dc_rel_ist_position*]

Direktbetrieb_TL ≙

Direktbetrieb_DC[*tl_geschwindigkeit_aktuell*/*dc_geschwindigkeit_aktuell*,
tl_rel_position_min/*dc_rel_position_min*,
tl_rel_position_max/*dc_rel_position_max*,
tl_rel_istposition/*dc_rel_ist_position*]

Direktbetrieb_CC ≙

Direktbetrieb_DC[*cc_geschwindigkeit_aktuell*/*dc_geschwindigkeit_aktuell*,
cc_rel_position_min/*dc_rel_position_min*,
cc_rel_position_max/*dc_rel_position_max*,
cc_rel_istposition/*dc_rel_ist_position*]

*//***
Änderung der Schrittweite

Die Eingabe eines neuen Wertes muss nicht mit "Enter" bestätigt werden.
Liegt der eingegebene Wert außerhalb eines motorspezifischen Intervalls,
so wird die aktuelle Schrittweite auf das Minimum bzw. das Maximum gesetzt.
***//*

Eingabe_Schrittweite_DC

Δ*Manuelle_Justage*

soll_schrittweite?: $\mathbb{Z}$

---

manuelle_justage = *Ein*

aktueller_antrieb = *DC*

aktuelle_antriebsart_dc = *schrittbetrieb*

if *soll_schrittweite?* < *dc_schrittweite_min*

then *dc_schrittweite_aktuell'* = *dc_schrittweite_min*

else if *soll_schrittweite?* > *dc_schrittweite_max*

then *dc_schrittweite_aktuell'* = *dc_schrittweite_max*

else *dc_schrittweite_aktuell'* = *soll_schrittweite?*

Eingabe_Schrittweite_DF $\widehat{=}$

Eingabe_Schrittweite_DC[*df_schrittweite_min*/*dc_schrittweite_min*,
df_schrittweite_aktuell'/*dc_schrittweite_aktuell'*,
df_schrittweite_max/*dc_schrittweite_max*]

Eingabe_Schrittweite_AR $\widehat{=}$

Eingabe_Schrittweite_DC[*ar_schrittweite_min*/*dc_schrittweite_min*,
ar_schrittweite_aktuell'/*dc_schrittweite_aktuell'*,
ar_schrittweite_max/*dc_schrittweite_max*]

Eingabe_Schrittweite_TL $\widehat{=}$

Eingabe_Schrittweite_DC[*tl_schrittweite_min*/*dc_schrittweite_min*,
tl_schrittweite_aktuell'/*dc_schrittweite_aktuell'*,
tl_schrittweite_max/*dc_schrittweite_max*]

Eingabe_Schrittweite_CC $\widehat{=}$

Eingabe_Schrittweite_DC[*cc_schrittweite_min*/*dc_schrittweite_min*,
cc_schrittweite_aktuell'/*dc_schrittweite_aktuell'*,
cc_schrittweite_max/*dc_schrittweite_max*]

*//***
Änderung der Geschwindigkeit

Die Eingabe muss nicht mit "Enter" bestätigt werden.
Liegt der eingegebene Wert außerhalb des motorspezifischen Intervalls,
so wird die jeweilige aktuelle Geschwindigkeit auf das Maximum gesetzt.
***//*

Eingabe_Geschwindigkeit_DC

Δ*Manuelle_Justage*
soll_geschwindigkeit?: $\mathbb{Z}$

---

manuelle_justage = *Ein*
aktueller_antrieb = *DC*
if *soll_geschwindigkeit?* $\in$ 0 .. *dc_geschwindigkeit_max*
then *dc_geschwindigkeit_aktuell'* = *soll_geschwindigkeit?*
$\wedge$ *dc_geschwindigkeit_aktuell_2'* = *soll_geschwindigkeit?*
else *dc_geschwindigkeit_aktuell'* = *dc_geschwindigkeit_max*
$\wedge$ *dc_geschwindigkeit_aktuell_2'* = *dc_geschwindigkeit_max*

Eingabe_Geschwindigkeit_DF $\widehat{=}$ *Eingabe_Geschwindigkeit_DC*[
df_geschwindigkeit_max/*dc_geschwindigkeit_max*,
df_geschwindigkeit_aktuell'/*dc_geschwindigkeit_aktuell'*,
f_geschwindigkeit_aktuell_2'/*dc_geschwindigkeit_aktuell_2'*]

Eingabe_Geschwindigkeit_AR $\widehat{=}$ *Eingabe_Geschwindigkeit_DC*[
ar_geschwindigkeit_max/*dc_geschwindigkeit_max*,
ar_geschwindigkeit_aktuell'/*dc_geschwindigkeit_aktuell'*,
r_geschwindigkeit_aktuell_2'/*dc_geschwindigkeit_aktuell_2'*]

Eingabe_Geschwindigkeit_TL $\widehat{=}$ *Eingabe_Geschwindigkeit_DC*[

tl_geschwindigkeit_max/dc_geschwindigkeit_max,
tl_geschwindigkeit_aktuell'/dc_geschwindigkeit_aktuell',
tl_geschwindigkeit_aktuell_2'/dc_geschwindigkeit_aktuell_2']

Eingabe_Geschwindigkeit_CC ≅ *Eingabe_Geschwindigkeit_DC*[
cc_geschwindigkeit_max/dc_geschwindigkeit_max,
cc_geschwindigkeit_aktuell'/dc_geschwindigkeit_aktuell',
cc_geschwindigkeit_aktuell_2'/dc_geschwindigkeit_aktuell_2']

*//***
Schrittbetrieb

Vorraussetzung ist der aktivierte Schrittbetrieb.
Das Ereignis "Links" oder "Rechts" trifft ein, wenn die linke bzw. rechte Pfeiltaste der Tastatur betätigt wird oder ein Mausklick auf das linke bzw. rechte Endeelement der Scrollbar erfolgt.
Der jeweilige aktivierte Antrieb bewegt sich mit der aktuellen Schrittweite nach links bzw. rechts.
Der Antrieb bewegt sich mit der aktuellen Geschwindigkeit, nicht wie gefordert mit der maximalen
Geschwindigkeit (Ausnahme: aktuelle Geschwindigkeit gleich Null, Bewegung in einem Sprung).
Falls sich der Antrieb an seiner maximalen Position befindet oder der Schritt den Motor über seine maximale Position hinaus bewegen würde, geschieht bei einem Schritt in diese Richtung nichts. (analog Minimum)
***//*

Schrittbetrieb_DC

Δ*Manuelle_Justage*

eingabe?: EREIGNISSE

---

manuelle_justage = Ein

aktueller_antrieb = DC

aktuelle_antriebsart_dc = schrittbetrieb

if *eingabe? = Links*

$\wedge$ *dc_rel_ist_position - dc_schrittweite_aktuell* $\geqslant$ *dc_rel_position_min*

then *dc_rel_ist_position' = dc_rel_ist_position - dc_schrittweite_aktuell*

if *eingabe? = Rechts*

$\wedge$ *dc_rel_ist_position + dc_schrittweite_aktuell* $\leqslant$ *dc_rel_position_max*

then *dc_rel_ist_position' = dc_rel_ist_position + dc_schrittweite_aktuell*

Schrittbetrieb_DF $\widehat{=}$

Schrittbetrieb_DC[*aktuelle_antriebsart_df*/*aktuelle_antriebsart_dc*,
df_rel_ist_position/*dc_rel_ist_position*,
df_schrittweite_aktuell/*dc_schrittweite_aktuell*,
df_rel_position_min/*dc_rel_position_min*,
df_rel_ist_position'/*dc_rel_ist_position'*,
df_rel_position_max/*dc_rel_position_max*]

Schrittbetrieb_AR $\widehat{=}$

Schrittbetrieb_DC[*aktuelle_antriebsart_ar*/*aktuelle_antriebsart_dc*,
ar_rel_ist_position/*dc_rel_ist_position*,
ar_schrittweite_aktuell/*dc_schrittweite_aktuell*,
ar_rel_position_min/*dc_rel_position_min*,
ar_rel_ist_position'/*dc_rel_ist_position'*,
ar_rel_position_max/*dc_rel_position_max*]

Schrittbetrieb_TL $\widehat{=}$

Schrittbetrieb_DC[*aktuelle_antriebsart_tl*/*aktuelle_antriebsart_dc*,
tl_rel_ist_position/*dc_rel_ist_position*,
tl_schrittweite_aktuell/*dc_schrittweite_aktuell*,
tl_rel_position_min/*dc_rel_position_min*,
tl_rel_ist_position'/*dc_rel_ist_position'*,
tl_rel_position_max/*dc_rel_position_max*]

Schrittbetrieb_CC ≙

Schrittbetrieb_DC[*aktuelle_antriebsart_cc*/*aktuelle_antriebsart_dc*,
cc_rel_ist_position/*dc_rel_ist_position*,
cc_schrittweite_aktuell/*dc_schrittweite_aktuell*,
cc_rel_position_min/*dc_rel_position_min*,
cc_rel_ist_position'/*dc_rel_ist_position'*,
cc_rel_position_max/*dc_rel_position_max*]

*//**
Fahrbetrieb

Analog zum Schrittbetrieb erfolgen die Eingaben "Links" und "Rechts".
Beim Fahrbetrieb wird der aktuelle Motor mit der aktuellen Geschwindigkeit so lange in die jeweilige Richtung bewegt, wie die entsprechende Eingabe erfolgt.
Die Bewegung endet an den jeweiligen maximal erreichbaren Positionen.
Ausnahme: aktuelle Geschwindigkeit gleich Null.
**//*

Fahrbetrieb_DC

Δ*Manuelle_Justage*
eingabe?: EREIGNISSE
laenge_der_eingabe: INTERVAL

---

manuelle_justage = Ein
aktueller_antrieb = DC
aktuelle_antriebsart_dc = fahrbetrieb
if *eingabe? = Links*
$\wedge$ *dc_rel_ist_position*
*- dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
< dc_rel_position_min
then *dc_rel_ist_position' = dc_rel_position_min*
else *dc_rel_ist_position'*
= dc_rel_ist_position
*- dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
$\vee$ (**if** *eingabe? = Rechts*
$\wedge$ *dc_rel_ist_position*
*+ dc_geschwindigkeit_aktuell * period laenge_der_eingabe*
> dc_rel_position_max
then *dc_rel_ist_position' = dc_rel_position_max*
else *dc_rel_ist_position'*
= dc_rel_ist_position
*+ dc_geschwindigkeit_aktuell * period laenge_der_eingabe*)
$\vee$ *dc_geschwindigkeit_aktuell* = 0
$\wedge$ *eingabe? = Links*
$\wedge$ *dc_rel_ist_position' = dc_rel_position_min*
$\vee$ *dc_geschwindigkeit_aktuell* = 0
$\wedge$ *eingabe? = Rechts*
$\wedge$ *dc_rel_ist_position' = dc_rel_position_max*

Fahrbetrieb_DF ≙

Fahrbetrieb_DC[*aktuelle_antriebsart_df*/*aktuelle_antriebsart_dc*,
df_geschwindigkeit_aktuell/*dc_geschwindigkeit_aktuell*,
df_rel_ist_position/*dc_rel_ist_position*,
df_rel_position_min/*dc_rel_position_min*,
df_rel_ist_position'/*dc_rel_ist_position'*,
df_rel_position_max/*dc_rel_position_max*]

Fahrbetrieb_AR ≙

Fahrbetrieb_DC[*aktuelle_antriebsart_ar*/*aktuelle_antriebsart_dc*,
ar_geschwindigkeit_aktuell/*dc_geschwindigkeit_aktuell*,
ar_rel_ist_position/*dc_rel_ist_position*,
ar_rel_position_min/*dc_rel_position_min*,
ar_rel_ist_position'/*dc_rel_ist_position'*,
ar_rel_position_max/*dc_rel_position_max*]

Fahrbetrieb_TL ≙

Fahrbetrieb_DC[*aktuelle_antriebsart_tl*/*aktuelle_antriebsart_dc*,
tl_geschwindigkeit_aktuell/*dc_geschwindigkeit_aktuell*,
tl_rel_ist_position/*dc_rel_ist_position*,
tl_rel_position_min/*dc_rel_position_min*,
tl_rel_ist_position'/*dc_rel_ist_position'*,
tl_rel_position_max/*dc_rel_position_max*]

Fahrbetrieb_CC ≙

Fahrbetrieb_DC[*aktuelle_antriebsart_cc*/*aktuelle_antriebsart_dc*,
cc_geschwindigkeit_aktuell/*dc_geschwindigkeit_aktuell*,
cc_rel_ist_position/*dc_rel_ist_position*,
cc_rel_position_min/*dc_rel_position_min*,
cc_rel_ist_position'/*dc_rel_ist_position'*,
cc_rel_position_max/*dc_rel_position_max*]

*//**
Setzen der relativen Null

Erfolgt die Eingabe "Setze_Relative_Null" wird für den aktuell gewählten Motor
die relative Null auf die aktuelle Position gesetzt (diese setzt sich aus der aktuellen relativen
Position und der aktuellen relativen Null zusammen)
Die aktuelle relative Position wird auf Null gesetzt.
**//*

```
┌──Relative_Null_Setzen_DC──────────────────────────────────────────
│  ΔXCTL
│  eingabe?: EREIGNISSE
├──────────────────────────
│  manuelle_justage = Ein
│  aktueller_antrieb = DC
│  eingabe? = Setze_Relative_Null
│  dc_rel_null' = dc_rel_ist_position + dc_rel_null
│  dc_rel_ist_position' = 0
└──────────────────────────────────────────────────────────────────
```

Relative_Null_Setzen_DF ≙
Relative_Null_Setzen_DC[*df_rel_null'*/*dc_rel_null'*,
df_rel_ist_position/*dc_rel_ist_position*,
df_rel_null/*dc_rel_null*,
df_rel_ist_position'/*dc_rel_ist_position'*]

Relative_Null_Setzen_AR ≙
Relative_Null_Setzen_DC[*ar_rel_null'*/*dc_rel_null'*,
ar_rel_ist_position/*dc_rel_ist_position*,
ar_rel_null/*dc_rel_null*,
ar_rel_ist_position'/*dc_rel_ist_position'*]

Relative_Null_Setzen_TL ≙
Relative_Null_Setzen_DC[*tl_rel_null'*/*dc_rel_null'*,
tl_rel_ist_position/*dc_rel_ist_position*,
tl_rel_null/*dc_rel_null*,
tl_rel_ist_position'/*dc_rel_ist_position'*]

Relative_Null_Setzen_CC ≙

Relative_Null_Setzen_DC[*cc_rel_null'*/*dc_rel_null'*,
cc_rel_ist_position/*dc_rel_ist_position*,
cc_rel_null/*dc_rel_null*,
cc_rel_ist_position'/*dc_rel_ist_position'*]

*//***
Aufheben der relativen Null

Die relative Null wird gleich der absoluten Null gesetzt.
Die aktuelle relative Position wird angepasst.
***//*

Aufheben_Relative_Null_DC

Δ*XCTL*
eingabe?: EREIGNISSE
manuelle_justage = Ein
aktueller_antrieb = DC
eingabe? = Relative_Null_Aufheben
dc_rel_null' = 0
dc_rel_ist_position' = dc_rel_ist_position + dc_rel_null

Aufheben_Relative_Null_DF ≙

Aufheben_Relative_Null_DC[*df_rel_null'*/*dc_rel_null'*,
df_rel_ist_position'/*dc_rel_ist_position'*,
df_rel_ist_position/*dc_rel_ist_position*,
df_rel_null/*dc_rel_null*]

Aufheben_Relative_Null_AR ≙

Aufheben_Relative_Null_DC[*ar_rel_null'*/*dc_rel_null'*,
ar_rel_ist_position'/*dc_rel_ist_position'*,
ar_rel_ist_position/*dc_rel_ist_position*,
ar_rel_null/*dc_rel_null*]

Aufheben_Relative_Null_TL ≙

Aufheben_Relative_Null_DC[*tl_rel_null'*/*dc_rel_null'*,
tl_rel_ist_position'/*dc_rel_ist_position'*,
tl_rel_ist_position/*dc_rel_ist_position*,
tl_rel_null/*dc_rel_null*]

Aufheben_Relative_Null_CC ≙

Aufheben_Relative_Null_DC[*cc_rel_null'*/*dc_rel_null'*,
cc_rel_ist_position'/*dc_rel_ist_position'*,
cc_rel_ist_position/*dc_rel_ist_position*,
cc_rel_null/*dc_rel_null*]

*//***
Verlassen von manuelle Justage

*Befindet sich ein oder mehrere Antriebe in Bewegung, so ist das Verlassen von Manuelle Justage (Alt) die einzige Möglichkeit den/die Antriebe zu stoppen. Die Position des jeweiligen Antriebs wird dann an das XCTL - Programm übergeben und hier durch die Variablen **_antrieb_ist_position dargestellt. (wahrscheinlich als direkte Information der Hardware, hier im Simulations-modus wird einfach die letzte Position ‚gemerkt', siehe XCTL-Verlassen). Die relative Null ist zu beachten, da sie im XCTL - Programm "gemerkt" wird. **//*

Manuelle_Justage_Verlassen

ΔXCTL
eingabe?: EREIGNISSE
dc_antrieb_ist_position?, df_antrieb_ist_position?, ar_antrieb_ist_position?,
tl_antrieb_ist_position?, cc_antrieb_ist_position?: ℤ

---

eingabe? = Verlassen_Manuelle_Justage
dc_rel_ist_position' = dc_antrieb_ist_position? - dc_rel_null
df_rel_ist_position' = dc_antrieb_ist_position? - df_rel_null
ar_rel_ist_position' = ar_antrieb_ist_position? - ar_rel_null
tl_rel_ist_position' = tl_antrieb_ist_position? - tl_rel_null
cc_rel_ist_position' = cc_antrieb_ist_position? - cc_rel_null

*//**
XCTL Verlassen

Beim Verlassen des XCTL - Programms werden die aktuellen Positionen (relativ zur absoluten Null) der Motoren im Ini_File (hardware.ini) gesichert.
***//*

XCTL_Verlassen

ΔINI_File
$XCTL$
$ereignis?: EREIGNISSE$

$ereignis? = Verlassen\_XCTL$
$dc\_ist\_position' = dc\_rel\_ist\_position + dc\_rel\_null$
$df\_ist\_position' = df\_rel\_ist\_position + df\_rel\_null$
$ar\_ist\_position' = ar\_rel\_ist\_position + ar\_rel\_null$
$tl\_ist\_position' = tl\_rel\_ist\_position + tl\_rel\_null$
$cc\_ist\_position' = cc\_rel\_ist\_position + cc\_rel\_null$

*//**
Die einzelnen Operationenwerden jeweils für die fünf Motoren zu einer Motorsteuerung zusammengefasst.
***//*

$Steuerung\_DC \cong$
$Auswahl\_Antriebsart\_DC$
$\vee\ Eingabe\_Schrittweite\_DC$
$\vee\ Eingabe\_Geschwindigkeit\_DC$
$\vee\ Direktbetrieb\_DC$
$\vee\ Schrittbetrieb\_DC$
$\vee\ Fahrbetrieb\_DC$
$\vee\ Relative\_Null\_Setzen\_DC$
$\vee\ Aufheben\_Relative\_Null\_DC$

$$
\begin{aligned}
&\mathit{Steuerung\_DF} \cong \\
&\quad \mathit{Auswahl\_Antriebsart\_DF} \\
&\quad \vee \mathit{Eingabe\_Schrittweite\_DF} \\
&\quad \vee \mathit{Eingabe\_Geschwindigkeit\_DF} \\
&\quad \vee \mathit{Direktbetrieb\_DF} \\
&\quad \vee \mathit{Schrittbetrieb\_DF} \\
&\quad \vee \mathit{Fahrbetrieb\_DF} \\
&\quad \vee \mathit{Relative\_Null\_Setzen\_DF} \\
&\quad \vee \mathit{Aufheben\_Relative\_Null\_DF}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Steuerung\_AR} \cong \\
&\quad \mathit{Auswahl\_Antriebsart\_AR} \\
&\quad \vee \mathit{Eingabe\_Schrittweite\_AR} \\
&\quad \vee \mathit{Eingabe\_Geschwindigkeit\_AR} \\
&\quad \vee \mathit{Direktbetrieb\_AR} \\
&\quad \vee \mathit{Schrittbetrieb\_AR} \\
&\quad \vee \mathit{Fahrbetrieb\_AR} \\
&\quad \vee \mathit{Relative\_Null\_Setzen\_AR} \\
&\quad \vee \mathit{Aufheben\_Relative\_Null\_AR}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Steuerung\_TL} \cong \\
&\quad \mathit{Auswahl\_Antriebsart\_TL} \\
&\quad \vee \mathit{Eingabe\_Schrittweite\_TL} \\
&\quad \vee \mathit{Eingabe\_Geschwindigkeit\_TL} \\
&\quad \vee \mathit{Direktbetrieb\_TL} \\
&\quad \vee \mathit{Schrittbetrieb\_TL} \\
&\quad \vee \mathit{Fahrbetrieb\_TL} \\
&\quad \vee \mathit{Relative\_Null\_Setzen\_TL} \\
&\quad \vee \mathit{Aufheben\_Relative\_Null\_TL}
\end{aligned}
$$

Steuerung_CC $\widehat{=}$

Auswahl_Antriebsart_CC
$\vee$ *Eingabe_Schrittweite_CC*
$\vee$ *Eingabe_Geschwindigkeit_CC*
$\vee$ *Direktbetrieb_CC*
$\vee$ *Schrittbetrieb_CC*
$\vee$ *Fahrbetrieb_CC*
$\vee$ *Relative_Null_Setzen_CC*
$\vee$ *Aufheben_Relative_Null_CC*

*//***
Operationen für Manuelle Justage
***//*

Steuerung_Manuelle_Justage $\widehat{=}$

Auswahl_Antrieb
$\vee$ *Motor_DC* $\wedge$ *Steuerung_DC* $\wedge \neg$ *Steuerung_DF*
$\vee$ *Motor_DF* $\wedge$ *Steuerung_DF* $\wedge \neg$ *Steuerung_DC*
$\vee$ *Motor_AR* $\wedge$ *Steuerung_AR*
$\vee$ *Motor_TL* $\wedge$ *Steuerung_TL*
$\vee$ *Motor_CC* $\wedge$ *Steuerung_CC*
$\vee$ *Manuelle_Justage_Verlassen*

A.2 Formale Spezifikation von „0-dimensionale Detektoren, Teilfunktion: Zählerfenster“

Hinweise zur Spezifikation

Die Spezifikation wurde kommentiert. Kommentare sind am Beginn durch „//**“ und am Ende durch „**//“ gekennzeichnet. Die Spezifikation wurde mit dem Editor der Software Z/Eves, in der Programmversion 2.1 erstellt.

*//** Das Zeitmodell*
Hundertstel $== \mathbb{N}$*, Hunderstel als Synonym für die narürlichen Zahlen*
Die Zeit als Schema.
Tick als Operationsschema der Zeit

***//*

$HUNDERTSTEL == \mathbb{N}$

Zeit

$t : HUNDERTSTEL$

Tick

$\Delta Zeit$

$t' = t + 1$

*//** Das Modell von 0-dimensionale Detektoren, Teilfunktion Zählerfenster*
***//*

*//** Definition freier Typen, Informationen aus den .ini Dateien und der verbalen Beschreibung**//*

DETEKTOR ::= *detektor* ⟪ℕ⟫

DETEKTORTYP ::= *RADICON* | *GENERIC* | *BRAUN_PSD* | *Test*
| *Simulant* | *PSD*

STATUS ::= *Ein* | *Aus*

XCONTROLSTARTUP ::= *Startup_Nothing*
| *Startup*
| *Startup_Scan*
| *Startup_AreaScan*
| *Startup_Device*
| *Startup_ManualAdjustment*
| *NIL*

ANZEIGE ::= *Digital* | *Balken*

*//** Eingaben die vom Nutzer getätigt werden können**//*

EREIGNISSE ::= *Verlassen_Zaehler_Konfiguration*
| *Verlassen_XCTL*
| *Verlassen_Zaehler_Fenster*
| *Abbrechen*
| *OK*
| *Markierung_Sound*
| *Markierung_Digitale_Anzeige*
| *Markierung_Balken_Darstellung*
| *Markierung_Fenster*
| *Markierung_Logarithmische_Dartsellung*
| *Markierung_Log_Datei*
| *Auswahl_Detektoren_Detektoren*
| *Auswahl_Einstellungen_Geraet*
| *Auswahl_Einstellungen_Anzeige*
| *Auswahl_Detektor*
| *Schließen*
| *Enter*

*//**
INI_File

In den Dateien Hardware.ini und XControl.ini sind diverse Parameter gegeben. Für die Teilfunktion Zählerfenster sind die Zeitbegrenzung und die Impulsbegrenzung für jeweils einen Detektor relevant. Die Parameter, die in Zusammenhang mit einem Detektor stehen, werden als Menge geordneter Paare dargestellt.

***//*

INI_File

$Detektoren: \mathbb{P}\ DETEKTOR$

$Zeitbegrenzungen\_Default: \mathbb{P}\ (DETEKTOR \times HUNDERTSTEL)$

$Impulsbegrenzungen\_Default: \mathbb{P}\ (DETEKTOR \times \mathbb{N})$

$Soundeinstellung\_Default: \mathbb{P}\ (DETEKTOR \times STATUS)$

$Detektortypen: \mathbb{P}\ (DETEKTOR \times DETEKTORTYP)$

$xcontrol\_startup: XCONTROLSTARTUP$

$default\_balkenanzahl: \mathbb{N}$

$min\_zeitbegrenzung, max\_zeitbegrenzung: HUNDERTSTEL$

$min\_impulsbegrenzung, max\_impulsbegrenzung: \mathbb{N}$

$min\_intensitaetsbegrenzung, max\_intensitaetsbegrenzung: \mathbb{N}$

*//**
XCTL-Basisprogramm
Hier finden sich alle Variablen, die bei aktivem XCTL-Programm verfügbar sind.
Die Variable sicherung_zeitbegrenzung wird benötigt, um die Änderung der aktuellen Zeitbegrenzung bei impulsbedingtem Neustart zu sichern.

***//*

XCTL

INI_File

zaehlerfenster, konfigurationsfenster: STATUS

Aktive_Detektoren: ℙ *DETEKTOR*

sound: STATUS

anzeige: ANZEIGE

aktueller_detektor: DETEKTOR

aktuelle_zeitbegrenzung, sicherung_zeitbegrenzung: HUNDERTSTEL

aktuelle_impulsbegrenzung: ℕ

*//***
Anzeige_Zaehlerfenster beschreibt das Verhalten, solange das Zählerfenster aktiv ist.
***//*

Anzeige_Zaehlerfenster

XCTL

aktuelle_intensitaet, default_intensitaet, min_intensitaet, max_intensitaet,

aktuelle_balkenanzahl, min_balkenanzahl, max_balkenanzahl: ℕ

anzeigeeinstellung, log_datei, pop_up_menu: STATUS

*//***
Der Zähler (Detektor).
***//*

Zaehler

XCTL

Zeit

aktueller_wert: ℕ

zeitbegrenung_2, beginn_zaehlung: HUNDERTSTEL

---

t - beginn_zaehlung ⩽ *aktuelle_zeitbegrenzung*

aktueller_wert ⩽ *aktuelle_impulsbegrenzung*

*//***
Initialisierung der Programmkomponenten XCTL, Anzeige_Zaehlerfenster, Zaehler

Auslesen der Parameter aus den .ini - Dateien. Welche Detektoren stehen zur Verfügung? (nur simulierte)
Eventuell öffnen des Zählerfensters, starten des Zählers.
***//*

Init_Zaehler

Zaehler'
ΞZeit

---

beginn_zaehlung' = t
aktueller_wert' = 0

Init_Anzeige_Zaehlerfenster

Anzeige_Zaehlerfenster'

---

aktuelle_intensitaet' = 140001
default_intensitaet' = 260000
min_intensitaet' = min_balkenanzahl' = 0
max_intensitaet' = 2000000
max_balkenanzahl' = 140
log_datei' = Aus

Init_XCTL

INI_File

XCTL'

Anzeige_Zaehlerfenster'

Zaehler'

$\Delta Zeit$

$\forall x: \mathbb{N}$

• $detektor\ x \in Aktive\_Detektoren'$

$\Leftrightarrow x = 0 \lor (\exists y: \mathbb{N} \mid detektor\ y \in Aktive\_Detektoren' \bullet x - y = 1)$

$Aktive\_Detektoren' \neq \varnothing$

$zaehlerfenster' = Aus \Leftrightarrow xcontrol\_startup = Startup\_Nothing$

$zaehlerfenster' = Ein \land Init\_Anzeige\_Zaehlerfenster \land Init\_Zaehler$

$\Leftrightarrow \neg\ xcontrol\_startup = Startup\_Nothing$

$konfigurationsfenster' = Aus$

$aktueller\_detektor' = detektor\ 0$

$\exists z: DETEKTOR \times HUNDERTSTEL \mid z \in Zeitbegrenzungen\_Default$

$\land\ z\ .\ 1 = detektor\ 0$

• $aktuelle\_zeitbegrenzung' = sicherung\_zeitbegrenzung' = z\ .\ 2$

$\exists i: DETEKTOR \times \mathbb{N} \mid i \in Impulsbegrenzungen\_Default \land i\ .\ 1 = detektor\ 0$

• $aktuelle\_impulsbegrenzung' = i\ .\ 2$

$\exists s: DETEKTOR \times STATUS \mid s \in Soundeinstellung\_Default$

$\land\ s\ .\ 1 = detektor\ 0 \bullet sound' = s\ .\ 2$

$t' = 0$

---

*//***

Definition der verschiedenen Operationen

***//*

Konfigurationsfenster_Einblenden

$\Delta XCTL$

$\Delta Anzeige\_Zaehlerfenster$

ereignis?: EREIGNISSE

---

ereignis? = Auswahl_Detektoren_Detektoren

konfigurationsfenster' = Ein

$\vee$ *ereignis? = Auswahl_Einstellungen_Geraet*

$\wedge$ *zaehlerfenster = Ein*

$\wedge$ *pop_up_menu = Ein*

$\wedge$ *pop_up_menu' = Aus*

$\wedge$ *konfigurationsfenster' = Ein*

Konfigurationsfenster_Ausblenden

$\Delta XCTL$

ereignis?: EREIGNISSE

---

konfigurationsfenster = Ein

ereignis? = OK $\vee$ *ereignis? = Abbrechen* $\vee$ *ereignis? = Enter*

konfigurationsfenster' = Aus

*//***

Speichern der Daten in einem Log-File.
Bei aktiviertem Zählerfenster kann man das Speichern aktivieren.
***//*

Messdaten_Speichern_Aktivieren

Δ*Anzeige_Zaehlerfenster*
ereignis?: EREIGNISSE

---

zaehlerfenster = Ein
konfigurationsfenster = Aus
pop_up_menu = Ein
pop_up_menu' = Aus
ereignis? = Markierung_Log_Datei
log_datei = Aus
log_datei' = Ein

Messdaten_Speichern_Deaktivieren

Δ*Anzeige_Zaehlerfenster*
ereignis?: EREIGNISSE

---

zaehlerfenster = Ein
konfigurationsfenster = Aus
pop_up_menu = Ein
pop_up_menu' = Aus
ereignis? = Markierung_Log_Datei
log_datei = Ein
log_datei' = Aus

*//***
Ändern der Anzeige des Zählerfensters: Digital oder Balkenanzeige.
***//*

Anzeige_Aendern

Δ*Anzeige_Zaehlerfenster*

ereignis?: EREIGNISSE

---

zaehlerfenster = Ein

konfigurationsfenster = Aus

anzeigeeinstellung = Aus

pop_up_menu = Ein

pop_up_menu' = Aus

$\wedge$ (*ereignis? = Markierung_Digitale_Anzeige* $\wedge$ *anzeige' = Digital*

$\vee$ *ereignis? = Markierung_Balken_Darstellung* $\wedge$ *anzeige' = Balken*)

*//***
Einblenden des Fensters zur Anzeigeeinstellung für die Änderung der Anzeigeart des Zählerfensters bei der Balkenansicht.
Voraussetzung ist ein aktiviertes Zählerfenster.
***//*

Anzeige_Einstellungen_Einblenden

Δ*Anzeige_Zaehlerfenster*

ereignis?: EREIGNISSE

---

zaehlerfenster = Ein

konfigurationsfenster = Aus

pop_up_menu = Ein

pop_up_menu' = Aus

anzeigeeinstellung = Aus

ereignis? = Auswahl_Einstellungen_Anzeige

anzeigeeinstellung' = Ein

Anzeige_Einstellungen_Ausblenden

Δ*Anzeige_Zaehlerfenster*

ereignis?: EREIGNISSE

---

zaehlerfenster = Ein

konfigurationsfenster = Aus

anzeigeeinstellung = Ein

ereignis? = Enter $\vee$ *ereignis? = OK* $\vee$ *ereignis? = Schließen*

anzeigeeinstellung' = Aus

*//***
Ändern der Anzeigeeinstellungen. Wird ein falscher Wert eingegeben, so wird nach Betätigung von 'OK' oder 'Enter' der eingegebene Wert korrigiert und der default - Wert eingetragen.
Das Fenster bleibt geöffnet.
Sonst: nach der Eingabe und 'OK' oder 'Enter' werden die Werte übernommen und das Fenster schließt sich.
***//*

Anzeige_Einstellungen_Aendern

Δ*Anzeige_Zaehlerfenster*
neue_intensitaet?, neue_balkenanzahl?: $\mathbb{N}$
ereignis?: EREIGNISSE
ereignis2?: EREIGNISSE

---

anzeigeeinstellung = Ein
if *min_intensitaet* $\leqslant$ *neue_intensitaet?* $\leqslant$ *max_intensitaet*
then *aktuelle_intensitaet' = neue_intensitaet?*
else (*ereignis2? = OK* $\vee$ *ereignis2? = Enter*)
$\wedge$ *aktuelle_intensitaet' = default_intensitaet*
if *min_balkenanzahl* $\leqslant$ *neue_balkenanzahl?* $\leqslant$ *max_balkenanzahl*
then *aktuelle_balkenanzahl' = neue_balkenanzahl?*
else (*ereignis2? = OK* $\vee$ *ereignis2? = Enter*)
$\wedge$ *aktuelle_balkenanzahl' = default_balkenanzahl*
ereignis? = OK $\vee$ *ereignis? = Enter*
anzeigeeinstellung' = Aus

*//***
Ein/Ausstellen der akustischen Ausgabe.
***//*

Aenderung_Sound

Δ*XCTL*
ereignis?: EREIGNISSE

---

konfigurationsfenster = Ein
ereignis? = Markierung_Sound
sound = Aus $\wedge$ *sound' = Ein* $\vee$ *sound = Ein* $\wedge$ *sound' = Aus*

*//**
Einblenden des Zählerfensters. Dies hat Auswirkungen auf den Zähler
und die Anzeigeparameter des Zählerfensters - der Zahler wird hier gestartet,
da nicht festgestellt werden konnte, wann genau er sonst startet.
***//*

Zaehlerfenster_Einblenden

ΔXCTL
ΔAnzeige_Zaehlerfenster
ΔZaehler
ereignis?: EREIGNISSE

---

konfigurationsfenster = Ein
ereignis? = Markierung_Fenster
zaehlerfenster = Aus
zaehlerfenster' = Ein
anzeige' = Digital
Init_Anzeige_Zaehlerfenster
Init_Zaehler

*//**
Ausblenden des Zählerfensters. Da simulierte Detektoren keine akustischen
Signale aussenden, konnte nicht geklärt werden, ob der Zählvorgang beim
Schließen des Zählerfensters endet.
Das Speichern der Daten in einer Datei endet.
***//*

Zaehlerfenster_Ausblenden

$\Delta XCTL$

$\Delta Anzeige\_Zaehlerfenster$

ereignis?: EREIGNISSE

---

konfigurationsfenster = Ein

ereignis? = Markierung_Fenster

zaehlerfenster = Ein

zaehlerfenster' = Aus

log_datei' = Aus

*//***
Auswahl des aktuellen Detektors. Dies hat Auswirkungen auf den Zähler und die Anzeigeoptionen des Zählerfensters.
***//*

Detektor_Auswahl

$\Delta XCTL$

$\Delta Anzeige\_Zaehlerfenster$

$\Delta Zaehler$

$neuer\_detektor?: DETEKTOR$

$z: DETEKTOR \times HUNDERTSTEL$

$i: DETEKTOR \times \mathbb{N}$

$s: DETEKTOR \times STATUS$

$ereignis?: EREIGNISSE$

$konfigurationsfenster = Ein$

$ereignis? = Auswahl\_Detektor$

$aktueller\_detektor' = neuer\_detektor?$

$anzeige' = Digital$

$log\_datei' = Aus$

$\wedge$ ($aktuelle\_zeitbegrenzung' = sicherung\_zeitbegrenzung' = z$. 2

$\Leftrightarrow z \in Zeitbegrenzungen\_Default \wedge z$. 1 $= neuer\_detektor?$)

$\wedge$ ($aktuelle\_impulsbegrenzung' = i$. 2

$\Leftrightarrow i \in Impulsbegrenzungen\_Default \wedge i$. 1 $= neuer\_detektor?$)

$\wedge$ ($sound' = s$. 2 $\Leftrightarrow s \in Soundeinstellung\_Default \wedge s$. 1 $= neuer\_detektor?$)

$Init\_Anzeige\_Zaehlerfenster$

$Init\_Zaehler$

*//***
Ändern der Zeitbegrenzung. Wird ein falscher Wert eingegeben, so wird nach Betätigung
von 'OK' oder 'Enter' der eingegebene Wert korrigiert und der default - Wert eingetragen.
Das Fenster schließt sich.
***//*

Aenderung_Zeitbegrenzung

$\Delta XCTL$

ΔINI_File

neue_zeitbegrenzung?: $\mathbb{N}$

ereignis?: EREIGNISSE

z: DETEKTOR $\times$ *HUNDERTSTEL*

---

konfigurationsfenster = Ein

if *min_zeitbegrenzung* $\leqslant$ *neue_zeitbegrenzung?* $\leqslant$ *max_zeitbegrenzung*

then *aktuelle_zeitbegrenzung' = sicherung_zeitbegrenzung'*

= neue_zeitbegrenzung?

else if *aktuelle_zeitbegrenzung' < min_zeitbegrenzung*

then *aktuelle_zeitbegrenzung' = sicherung_zeitbegrenzung'*

= min_zeitbegrenzung

else *aktuelle_zeitbegrenzung' = sicherung_zeitbegrenzung'*

= max_zeitbegrenzung

ereignis? = Enter $\vee$ *ereignis? = OK*

konfigurationsfenster' = Aus

---

*//***

Ändern der Impulsbegrenzung. Wird ein falscher Wert eingegeben, so wird nach Betätigung
von 'OK' oder 'Enter' der eingegebene Wert korrigiert und der default - Wert eingetragen.
Das Fenster schließt sich.
***//*

Aenderung_Impulsbegrenzung

$\Delta XCTL$

ΔINI_File

neue_impulsbegrenzung?: $\mathbb{N}$

ereignis?: EREIGNISSE

i: DETEKTOR $\times \mathbb{N}$

---

konfigurationsfenster = Ein

if *min_impulsbegrenzung* $\leqslant$ *neue_impulsbegrenzung?* $\leqslant$ *max_impulsbegrenzung*

then *aktuelle_impulsbegrenzung' = neue_impulsbegrenzung?*

else if *aktuelle_impulsbegrenzung' < min_impulsbegrenzung*

then *aktuelle_impulsbegrenzung' = min_impulsbegrenzung*

else *aktuelle_impulsbegrenzung' = max_impulsbegrenzung*

ereignis? = Enter $\vee$ *ereignis? = OK*

konfigurationsfenster' = Aus

*//***
Darstellung des Zählvorgangs.
input? ist die Information, dass ein Photon festgestellt wurde.
***//*

Zaehlvorgang

$\Delta Zaehler$

input?: $\mathbb{N}$

---

aktueller_wert' = aktueller_wert + input?

*//***
Abbruch des Zählvorgangs beim Erreichen der Zeitbegrenzung, Ausgabe des Wertes und Neustart des Zählers.
Die aktuelle Zeitbegrenzung wird neu eingelesen, da sie durch einen impulsbedingten Zählerneustart verändert sein könnte.
***//*

Zaehler_Neustart_Zeitbedingt

$\Delta Zaehler$

$\Xi Zeit$

output!: $\mathbb{N}$

z: DETEKTOR $\times \mathbb{N}$

---

t - beginn_zaehlung > aktuelle_zeitbegrenzung

aktueller_wert' = 0

beginn_zaehlung' = t

output! = aktueller_wert

aktuelle_zeitbegrenzung' = sicherung_zeitbegrenzung

*//***
Abbruch des Zählvorgangs beim Erreichen der Impulsbegrenzung, Neustart des Zählers bei geringerer Zeitbegrenzung.
***//*

Zaehler_Neustart_Impulsbedingt

$\Delta Zaehler$

$\Xi Zeit$

output!: $\mathbb{N}$

---

aktueller_wert > aktuelle_impulsbegrenzung

output! = aktueller_wert

aktuelle_zeitbegrenzung' = aktuelle_zeitbegrenzung - (t - beginn_zaehlung)

$\wedge$ *beginn_zaehlung' = t*

aktueller_wert' = 0

*** / Beim Verlassen des XCTL-Programms werden die aktuellen Zeitbegrenzungs-,*
Impulsbegrenzungs- und Soundeinstellungen in der .ini-Datei gesichert.
*/***

XCTL_Verlassen

ΔINI_File

$XCTL$

$z: DETEKTOR \times HUNDERTSTEL$

$i: DETEKTOR \times \mathbb{N}$

$s: DETEKTOR \times STATUS$

$Zeitbegrenzungen\_Default'$

$= Zeitbegrenzungen\_Default \setminus \{z\}$

$\cup \{(aktueller\_detektor, aktuelle\_zeitbegrenzung)\}$

$\Leftrightarrow z \in Zeitbegrenzungen\_Default \land z \,.\, 1 = aktueller\_detektor$

$Impulsbegrenzungen\_Default'$

$= Impulsbegrenzungen\_Default \setminus \{i\}$

$\cup \{(aktueller\_detektor, aktuelle\_impulsbegrenzung)\}$

$\Leftrightarrow i \in Impulsbegrenzungen\_Default \land i \,.\, 1 = aktueller\_detektor$

$Soundeinstellung\_Default'$

$= Soundeinstellung\_Default \setminus \{s\} \cup \{(aktueller\_detektor, sound)\}$

$\Leftrightarrow s \in Soundeinstellung\_Default \land s \,.\, 1 = aktueller\_detektor$

A.3 Überarbeitete Version von „Verhaltensspezifikation (Pflichtenheft) XCTL-Steuerprogramm Funktion: Probe und Kollimator manuell justieren“ [4]

Dokumentversion: 2.2 (4.7.2001), erweitert durch Jörg Lange
Autor: K. Bothe
Zustand: in Bearbeitung
Änderungen zur Version 1.1:
- neues erweitertes Gliederungsschema
- Ergebnisse des Reviews im Seminar 'SW-Sanierung' (WS00/01)

Erweiterungen:

- Ungenauigkeiten sind blau markiert
- Fehler sind rot markiert
- Ergänzungen sind grün markiert

Gliederung

0. Gültigkeitsbereich des Dokumentes

Das Dokument bezieht sich ausschließlich auf den Bereich der Topographie. Eine Verallgemeinerung des Dokumentes hinsichtlich der Diffraktometrie /Reflektometrie und sonstiger Meßplätze steht noch aus.

Topographiespezifisch sind insbesondere die Anzahl und die Bezeichnungen der Motoren, sowie das Auswahlfeld 'Antrieb-Schnellwahl'. Des Weiteren sind einige der aufgeführten Werte aus dem ini-File spezifisch für den Topographiearbeitsplatz RTK4.

Von hohem Allgemeinheitsgrad sind die Punkte 2.3 und 2.4.

1. Überblick

Aufgabe der Funktion *Probe und Kollimator manuell justieren* ist es, eine Probe und einen Kollimator im dreidimensionalen Raum für ein physikalisches Experiment (Topographie) zu positionieren.

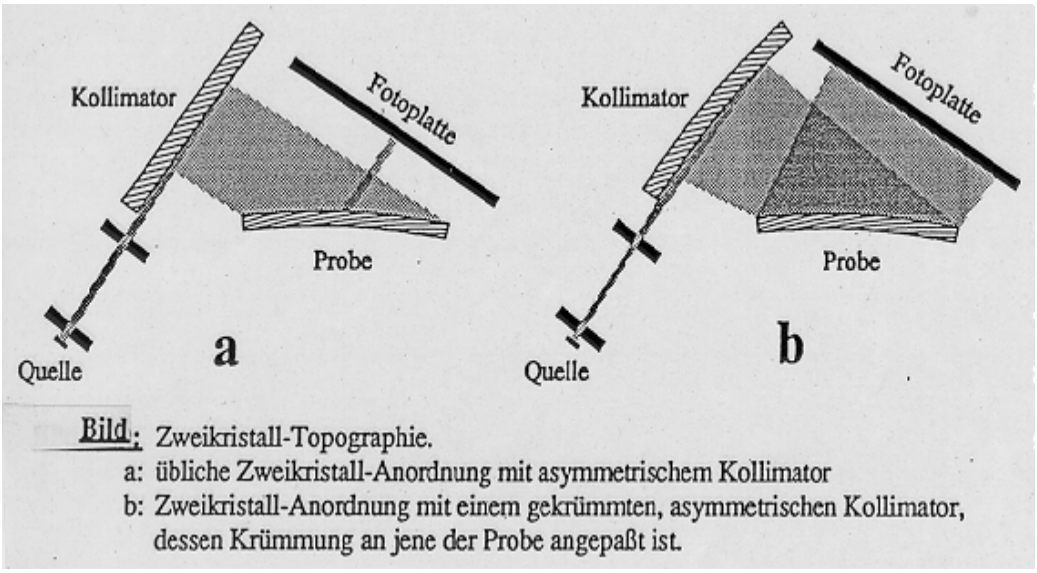


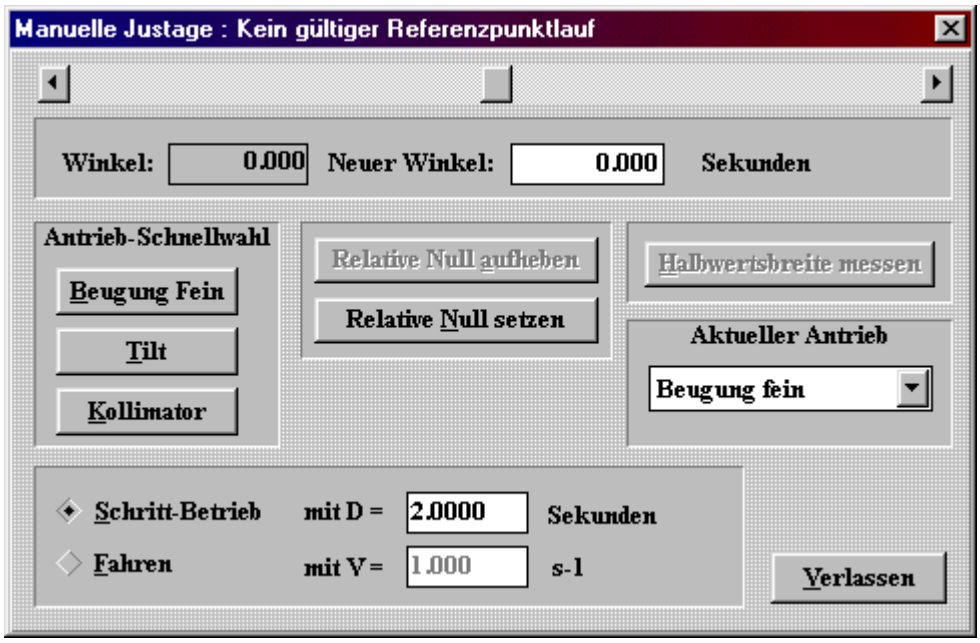
Bild: Zweikristall-Topographie.
a: übliche Zweikristall-Anordnung mit asymmetrischem Kollimator
b: Zweikristall-Anordnung mit einem gekrümmten, asymmetrischen Kollimator, dessen Krümmung an jene der Probe angepaßt ist.

Für die Positionierung der Probe bei der *Topographie* stehen vier unterschiedliche Antriebe zur Verfügung, die eine Bewegung der Probe in verschiedenen Richtungen ermöglichen. Zusätzlich krümmt ein weiterer Antrieb den Kollimator (zur Ablenkung des Röntgenstrahls). Die Positionierung erfolgt schrittweise durch manuelle Einstellung über eine Dialogbox, die zur Einzelbewegung der Motoren führt. Die Güte der aktuellen Einstellung wird akustisch und/oder optisch (Detektorsignale) und über die Messung der Halbwertsbreite kontrolliert. Nach Ausführung dieser Funktion wird der

Vorgang der Topographie über separate Dialogboxen (siehe dort) angestoßen.

2. Funktionale Beschreibung

Dialogbox:



2.1 Antriebe zur Positionierung der Probe

Für die Positionierung der Probe bei der **Topographie** stehen vier unterschiedliche Antriebe zur Verfügung, die eine Bewegung der Probe in verschiedenen Richtungen ermöglichen. Es handelt sich um die Antriebe (synonym: Motoren)

- DC (Diffraction angle coarse, Grobeinstellung des Beugungswinkels; Beugung grob),
- DF (Diffraction angle fine, Feineinstellung des Beugungswinkels; Beugung fein),
- AR (Azimuthal Rotation, Azimutale Rotation) und
- TL (Tilt, Verkippung).

Zusätzlich existiert ein weiterer Antrieb zur Krümmung des Kollimators (Halbleiterplättchen zur Ablenkung und Auffächerung des Röntgenstrahls):

- CC (Collimator-Curvature, Kollimator-Krümmung).

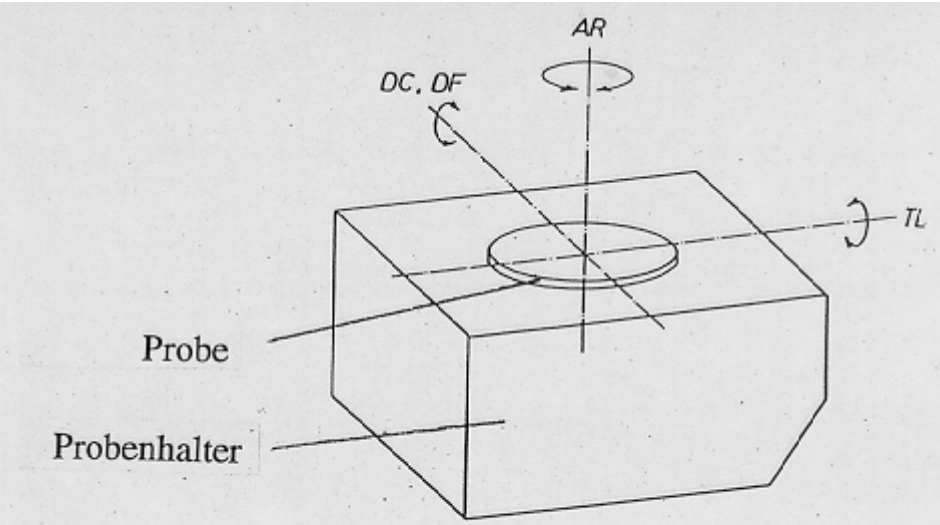


Bild: Freiheitsgrade einer Probe auf einem Probenhalter

Über die Dialogbox kann ein 'aktueller Antrieb' aus den fünf oben genannten ausgewählt werden (entweder durch 'Antrieb-Schnellwahl' oder durch die Klappliste 'Aktueller Antrieb'. Alle Werte in der Dialogbox beziehen sich dann auf diesen aktuellen Antrieb, wodurch zu einem Zeitpunkt immer nur ein Antrieb durch die Dialogbox gesteuert werden kann.

2.2 Anwendungsszenarium

Die Antriebe (Motoren) werden unabhängig voneinander bedient und können gleichzeitig in Bewegung sein. Lediglich die Antriebe DF und DC können sich nicht gleichzeitig bewegen, da sie in derselben Bewegungsrichtung wirken. Eine gleichzeitige Bewegung zweier Motoren wird wie folgt ausgelöst: Der aktuelle Motor wird in Bewegung gesetzt. Während sich dieser in Bewegung befindet, wird ein anderer Motor ausgewählt und bei diesem ebenfalls eine Bewegung ausgelöst.

Befindet sich ein Antrieb in Bewegung, so werden weitere Benutzereingaben für diesen Antrieb ignoriert, bis die Bewegung abgeschlossen wurde. Der Benutzer wird darüber mit einer Warnung "Motor <Name des Motors> in Bewegung" informiert (#Frage: wirklich in derzeitiger Version?) (#Wunsch: besser: erst keine Eingabe ermöglichen). Eingaben zu anderen Antrieben sind (wie oben beschrieben) möglich.

Vor jedem physikalischen Experiment werden mit Hilfe der genannten Antriebe die Probe positioniert und die Krümmung des Kollimators eingestellt. Hierzu werden vom Benutzer über die Dialogbox *Manuelle Justage*für jeden Antrieb

die einzustellenden Positionen (Soll-Positionen) angesteuert. Je nach gewählter Betriebsart (s. u.) werden die Antriebe in unterschiedlicher Weise von ihrer aktuellen Position (Ist-Position) auf die Soll-Position gefahren. Die Soll-Positionen hängen von der jeweiligen Probe ab und werden erst im Laufe der Einstellung ermittelt.

Zunächst werden die Antriebe DC und AR eingestellt. Danach im Wechsel (iterativer Prozeß) die drei anderen Antriebe: DF, TL, CC (deshalb als Schnellauswahl im Dialogfenster bereitgestellt). Der iterative Prozeß wird ca. 20 - 100 Mal wiederholt, bis die gemessene Roentgenstrahlung dem Maximalwert möglichst nahe kommt. Dies kann mit einer Messung der Halbwertsbreite (s. 2.4 und im Glossar) kontrolliert werden. Je geringer die Halbwertsbreite ist, desto besser ist die Probe eingestellt.

2.3 Bewegung der einzelnen Antriebe

Die folgenden Angaben treffen auf jeden Antrieb zu:

a. **Aktuelle Position**

   Jeder Antrieb besitzt eine aktuelle Position (Ist-Position), die dem Anwender in einem Feld der Dialogbox (Anzeigefeld *Winkel*) als Zahlenwert und durch die Position eines Schiebereglers in horizontaler Ausrichtung angezeigt wird. Die aktuelle Position wird bei einem sich bewegendem Motor kontinuierlich verändert. Die graphische Benutzungsoberfläche des Systems stellt sicher, daß Zahlenwert der aktuellen Position des Antriebs und die Position des Schiebereglers stets miteinander übereinstimmen. Alle Angaben (Zahlenwerte, Schieberegler) beziehen sich auf die relative Null (siehe: b). (#Anmerkung: Ist das sinnvoll, d. h. eine Anzeige bzgl. der absoluten Null besser?).

   Anmerkung: Der absolute Nullpunkt eines Antriebs wird vor Nutzung der 'Manuellen Justage' mittels eines Referenzpunktlaufes über die Dialogbox 'Reference Point Handling' ermittelt. Dieser entspricht per definitionem dem Ursprung des verwendeten Koordinatensystems der absoluten Positionen. Der absolute Nullpunkt liegt im Allgemeinen mittig zwischen der physikalisch minimalen und maximalen Position.

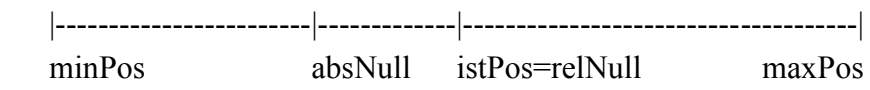
   Ausnahme:
   Text im Kopf der Dialogbox: Gibt Hinweis auf die Gültigkeit der angezeigten Ist-Postion [Anzeigefeld: Winkel] nach der Initialisierung des Systems (Programmstart):

- "Manuelle Justage" bedeutet: angezeigte Ist-Position ist korrekt
- "Manuelle Justage: Kein gueltiger Referenzpunktlauf" bedeutet: Aktuell angezeigte Position [Winkel] ist nicht zutreffend für die reale Position, da zuletzt ein Programmabsturz erfolgt war (Position konnte in diesem Fall nicht gerettet werden).

b. **Relative Null**

Alle Winkelangaben der Dialogbox beziehen sich auf die relative Null-Position. Initial entspricht die relative Null-Position der absoluten Null. Es kann aber jede beliebige Ist-Position durch das Eingabefeld *relative Null setzen* zur relativen Null erklärt werden. Durch das Eingabefeld *relative Null aufheben* wird die relative Null-Position wieder auf die absolute Null zurückgesetzt, durch erneutes *relative Null setzen* kann (auch ohne vorheriges Aufheben der bisherigen relativen Null) eine andere Position zur relativen Null erklärt werden.

Bsp.: Die Ist-Position liegt oberhalb der abs.Null und wird zur rel.Null erklaert.



Anm.: Das Setzen der relativen Null hat eine Auswirkung auf den als 'Winkel' bzw. 'Neuer Winkel' gültigen Wertebereich. Wenn z.B. die relative Null auf den minimalen Winkel gesetzt wurde, können nur noch positive Winkelpositionen auftreten. Die relative Null kann nicht bei laufendem Motor gesetzt werden. Beim Setzen der relativen Null wird der Schieberegler auf die Mitte gesetzt. Beim Aufheben der relativen Null wird wieder der Absolutwert der aktuellen Position angezeigt. Der Schieberegler wird entsprechend gesetzt.

c. **Betriebsart und Start der Bewegung**

Für jeden Antrieb gibt es drei Betriebsarten, um die gewünschte Soll-Position anzusteuern:

1. Direktbetrieb: Die Soll-Position wird im Eingabefeld *Neuer Winkel* als Absolutwert angegeben. Der Antrieb fährt in die gewünschte Soll-Position. Die Bewegung wird (derzeit) durch Drücken der 'Enter'-Taste ausgelöst (#Anmerkung: ergonomisch

schlecht, da diese Betriebsart, im Gegensatz zu den anderen Arten - in der Dialogbox nicht explizit ausgewiesen ist).

2. Schrittbetrieb: Die Soll-Position wird relativ zur Ist-Position durch die Eingabe einer Schrittweite im Eingabefeld *D=* definiert. Die Bewegung wird aktiviert, wenn eine der Cursortasten der Tastatur (<- oder ->) betätigt wird bzw. auf dem Scrollbar das linke oder rechte Endeelement angeklickt wird. Durch Betätigen der Cursortaste -> bzw. Scrollbar 'rechtes Endeelement' (u. U. ergonomisch problematisch, da kein extra Start-Knopf) fährt der Antrieb bis zur Position *Ist-Position + Schrittweite*. Durch Betätigen der Cursortaste <- bzw. Scrollbar 'linkes Endeelement' wird der Antrieb auf die Position *Ist-Position - Schrittweite* bewegt.

3. Fahrbetrieb: Der Antrieb wird solange bewegt, wie die Cursortasten (← oder →) der Tastatur oder die Endeelemente des Scrollbars gedrückt gehalten werden. Die Bewegungsrichtung richtet sich nach der betätigten Taste. Mit der Cursortaste -> bzw. Scrollbar 'rechtes Endeelement' wird ein größerer Winkel (= größere Position) eingestellt, mit der Cursortaste <- bzw. Scrollbar 'linkes Endeelement' wird der Winkel (die Position) verkleinert.

→ Ungenauigkeit

Für die Betriebsarten Schrittbetrieb und Fahrbetrieb wird der Antrieb aktiviert, wenn die linke oder rechte Pfeiltaste oder das linke oder rechte Endelement der Laufleiste (Scrollbar) betätigt werden. Es fehlt der Hinweis, dass die jeweilige Antriebsart aktiviert / markiert werden muss, falls dies nicht so ist, bevor die Bewegung gestartet wird.

→ Beobachtung

Wird bei aktiviertem Schrittbetrieb eine Richtungstaste bzw. das entsprechende Endelement der Laufleiste dauerhaft betätigt, erfolgen mehrere aufeinander folgende Schritte. Zum Verhältnis zwischen Dauer und Anzahl der Schritte kann keine genaue Aussage getroffen werden

a. **Bewegungsgeschwindigkeit**

In den Betriebsarten *Direktbetrieb* und *Fahrbetrieb* kann die Bewegungsgeschwindigkeit des Antriebs vorgegeben werden. Sie kann vom Benutzer im Eingabefeld *Fahren mit V = ... s-1* eingetragen werden. Im *Schrittbetrieb* wird der Antrieb immer mit der maximal möglichen Geschwindigkeit bewegt (hier liegt der Gedanke zu Grunde, daß in sehr kleinen Schritten bewegt wird, die mehrfach wiederholt werden sollen).

→ Fehler

Hier wurde ein Fehler in der Verhaltensbeschreibung entdeckt. Die Geschwindigkeit mit der sich ein Antrieb im Schrittbetrieb bewegt, ist unter bestimmten Vorraussetzungen nicht maximal.

Fall 1: Schrittbetrieb aktiviert

Wird eine neue zulässige Bewegungsgeschwindigkeit eingegeben, so gilt diese ebenfalls für den Direktbetrieb. Wird nach der Eingabe oder erst nach einer Bewegung im Direktbetrieb der Schrittbetrieb aktiviert, bewegt sich der Antrieb mit der eingegebenen Geschwindigkeit.

Erfolgt ein Wechsel in den Fahrbetrieb, bleibt die ausgewählte Geschwindigkeit bestehen (korrekt). Erfolgt ein Wechsel zurück in den Schrittbetrieb, wird der Geschwindigkeitswert auf sein Maximum gesetzt (korrekt für Schrittbetrieb). Dies hat jedoch eine Auswirkung auf den Direktbetrieb.

Fall 2: Fahrbetrieb aktiviert

Wird eine neue zulässige Bewegungsgeschwindigkeit eingegeben, so gilt diese für den Fahrbetrieb und den Direktbetrieb. Erfolgt ein Wechsel in den Schrittbetrieb, wird die Geschwindigkeit auf das Maximum gesetzt. Dies hat auch Auswirkungen auf den Direktbetrieb (siehe Fall 1). Erfolgt ein Wechsel zurück in den Fahrbetrieb, wird die Geschwindigkeit auf den zuvor eingegebenen Wert gesetzt.

→ Beobachtung zu Bewegungsgeschwindigkeit gleich Null

Erfolgt in dem Geschwindigkeitsfeld die Eingabe 0, wird ein nicht erwartetes Verhalten des Systems beobachtet.

Fall 1 Direktbetrieb wird aktiviert / ausgeführt

Nachdem die Eingabe einer neuen Position in das Feld ‚Neuer Winkel' erfolgt ist und der Direktbetrieb mit ‚Enter' aktiviert wird, geschieht optisch nichts. Wird eine Einstellung vorgenommen, beispielsweise die Antriebsart gewechselt, erfolgt ein ‚Sprung' zur eingegebenen Position.

Fall 2 Fahrbetrieb wird aktiviert / ausgeführt

Wird der Fahrbetrieb durch Betätigung einer Richtungstaste aktiviert, geschieht wie in Fall 1 optisch nichts. Wird eine Einstellung vorgenommen, erfolgt auch hier ein ‚Sprung'. Je nach dem, ob die linke oder rechte Bewegung ausgelöst wurde, zur minimalen oder maximalen Position.

Fall 3 Schrittbetrieb

Wird der Schrittbetrieb aktiviert entspricht die „Sprungweite" der aktuellen Schrittweite. Die Richtung ist abhängig von der Eingabe die den Schrittbetrieb aktiviert hat. Die Veränderung wird auch hier erst nach einer weiteren Einstellung sichtbar.

Um dieses Problem zu umgehen, könnte man das Geschwindigkeitsminimum größer Null festlegen. Eine Bewegung mit Geschwindigkeit gleich Null ist nicht sinnvoll.

b. **Reaktion auf unzulässige Eingaben**

Die Antriebe müssen durch die Software vor unzulässigen Eingaben geschützt werden. (#Wunsch: Sinnvoll, jedoch derzeit nicht realisiert: Bei unzulässigen Eingaben wird der Benutzer durch einen Warnton akustisch gewarnt)

1. *Unzulässige Sollposition:* Wird im *Direktbetrieb* eine Soll-Position eingegeben, die außerhalb des zulässigen Wertebereichs liegt, so wird die Positionsangabe auf die minimal oder maximal zulässige Position korrigiert, je nachdem, ob der zulässige Wertebereich unterschritten oder überschritten wird. Im *Fahrbetrieb* kann der Antrieb ebenfalls nur innerhalb des zulässigen Wertebereichs bewegt werden. Eine Unter- oder Überschreitung wird in diesem Fall durch Stoppen des Antriebs verhindert.

2. *Unzulässige Bewegungsgeschwindigkeit:* Wird im *Direktbetrieb* oder im *Fahrbetrieb* eine unzulässig große Bewegungsgeschwindigkeit eingegeben, so wird die Bewegungsgeschwindigkeit auf die maximal mögliche Bewegungsgeschwindigkeit gesetzt.

3. *Unzulässige Schrittweite:* Wird im *Schrittbetrieb* für die Schrittweite ein Wert eingegeben, der außerhalb des zulässigen Wertebereichs für die Schrittweite liegt, wird die Schrittweite auf den minimal oder maximal möglichen Wert gesetzt, je nachdem, ob der zulässige Wertebereich unterschritten oder überschritten worden ist. Falls durch die Nutzung einer zulässigen Schrittweite der Bereich für zulässige Winkelpositionen verlassen wird, muß ebenfalls eine Korrektur der Schrittweite vorgenommen werden.

→ Beobachtung

Wird oder ist eine zulässige Schrittweite eingegeben mit der durch einen Schritt die zulässige Winkelposition verlassen werden würde, so erfolgt bei der Aktivierung des Schrittbetriebes keine Reaktion (entsprechende Bewegungsrichtung vorausgesetzt).

(#Wunsch: Eine Korrektur von unzulässigen Eingaben durch die Verwendung von Grenzwerten sollte durch den Benutzer bestätigt werden.)

c. **Stoppen der Bewegung**

1. Direktbetrieb: Nur durch den Knopf 'Verlassen' der Dialogbox 'Manuelle Justage' (#Anmerkung: ergonomisch sehr schlecht)

2. Schrittbetrieb: Nur durch den Knopf 'Verlassen' der Dialogbox 'Manuelle Justage' (#Anmerkung: ergonomisch sehr schlecht)

3. Fahrbetrieb: Cursor-Taste loslassen

Damit werden beim 'Verlassen' des Dialogs immer die laufenden Antriebe gestoppt. Die Position wird gemerkt. (#Anmerkung: Ergonomisch ist die Lösung problematisch. Besser wäre beim Drücken dieser Taste im Falle von sich noch bewegenden Antrieben eine Nachricht und eine Frage 'Nicht alle Antriebe stehen', 'Sind Sie sicher?'.) Anmerkung: Es existiert ein Hardware-Kommando 'Abort Motion' zum

Stoppen eines Motors, das intern (also beim "Verlassen" des Dialogs) benutzt wird.

2.4 Messung der Halbwertsbreite

Mit der Betätigung von [Halbwertsbreite messen] wird die Messung der Halbwertsbreite für die aktuelle Position der Probe ausgelöst. Nach dem Abschluss der Messung erscheint eine Message-Box mit dem Ergebnis.

3. Daten

3.1 Zusammenfassung

Eine Beeinflussung des Systemverhaltens ist durch Daten aus der Dialogbox *und* des ini-Files möglich.

Über die Dialogbox 'Manuelle Justage' kann eine Reihe von Daten zur Motorbewegung eingegeben werden:

- Auswahl des Antriebs (DF, DC, TL, AR, CC),
- Betriebsart (Direktbetrieb, Schrittbetrieb, Fahrbetrieb),
- Soll-Position (Eingabefeld *Neuer Winkel*),
- Schrittweite (Eingabefeld *D*=),
- Bewegungsrichtung (Betätigen der Cursortasten <- oder ->),
- Bewegungsgeschwindigkeit (Eingabefeld *Fahren mit V = ... s-1*),
- relative Null setzen, relative Null löschen.

Diese Daten werden durch das System zu den entsprechenden Daten aus dem ini-File in Beziehung gesetzt, in denen Parameter zu den Motoren stehen. Jeder Arbeitsplatz besitzt hinsichtlich seiner Geräte ein spezifisches ini-File.

3.2 Ini-File

Eine Reihe von Angaben des ini-Files bestimmt charakteristische Parameter der Motoren. Eine vollständige Aufzählung der motorspezifischen Parameter findet sich im Dokument 'Verhaltensspezifikation: XCTL-Steuerprogramm, Teil: Motorsteuerungs-Komponente (motors.dll)', zu finden unter: IX Entwicklerdokumente, Analyse und Definition, Motoren, Verhaltens-spezifikation v.0.7, 3. Daten. Diese Werte werden durch den Nutzer gesetzt. Hierfür gibt es bspw. die Dialogbox 'Motor-Parameter'. Zu Beginn einer Sitzung (Programmstart) wird das ini-File gelesen, wodurch eine Steuerung des Dialogs über die im ini-File gesetzten Werte erfolgt.

Anmerkung: Neben der Möglichkeit, ini-File-Parameter über Dialogboxen zu setzen, kann der Anwender das ini-File auch direkt editieren, wovon tatsächlich Gebrauch gemacht wird. Die dadurch hervorgerufenen unkontrollierten Veränderungen stellen eine kritische Fehlerquelle dar.

3.3 Zulässige Motorpositionen (absolute Positionen)

Für die Antriebe DC, DF und TL erfolgt die Eingabe der Soll-Position in Winkelsekunden (1 Grad = 3.600 Winkelsekunden), für den Antrieb AR in Grad und für den Antrieb CC in Mikrometer. Die zulässigen Wertebereiche finden sich als Konstanten im ini-File: AngleMin, AngleMax für den jeweiligen Antrieb.
Die Werte im ini-File entsprechen nicht unbedingt der physisch maximalen Position! Falls sie die physisch maximale Position überschreiten, wird das Programm diesen Spezialfall abfangen. Außerdem gibt es zwei Konstanten im ini-File für die Auflösungsweite: AngleWidth, PositionWidth. (#Frage: Eine - welche - ist für die Darstellung in der Scrollbar bestimmt).

→ Beobachtung

Zu Beginn einer Sitzung entspricht die hier benannte Konstante AngleWidth dem in der Dialogbox angezeigten Wert für die Schrittweite. (In dem folgenden Beispiel wird die blau markierte Frage in 3.3 beantwortet, AngleWidth als Auflösung (Schrittweite).)

Beispiel für Konstanten AngleMin, AngleMax, AngleWidth eines speziellen ini-Files:

- DC [-15.399,4 .. 15.815,6], Auflösung 0,4 Sekunden,
- DF [-1.255,6 .. 1.255,6], Auflösung 0,02 Sekunden,
- TL [-18.990,38 .. 16.211,30], Auflösung 0,5 Sekunden,
- AR [-296,91 .. 296,04], Auflösung 0,001 Grad,
- CC [-1.063,8 .. 1.063,8], Auflösung 0,12 Mikrometer.

#Anmerkung: bei DC und TL in obigem Beispiel: Wertebereich kein Vielfaches der Auflösung (Maximalposition nicht erreichbar)! (#Frage: zur Anmerkung: Gilt dies auch bei Verwendung der minimalen Schrittweite?)

Anmerkung:
Ist-Positionen und Soll-Positionen in den Eingabefeldern 'Winkel' und 'neuer Winkel' werden als relative Positionen (bzgl. der relativen Null) eingetragen. Damit hat das Setzen der relativen Null Auswirkungen auf den Wertebereich der relativen Positionen für die Wahl der Winkel-Angaben.

Anmerkung:
Winkelangaben des Nutzers werden intern in Motor-Encoderschritte umgeformt. Ein Encoderschritt ist die minimale Bewegung, die ein Motor (physisch) ausführen kann. Damit ist mit einer Transformation von eingegebnen Winkeln eine Rundung in ganzzahlige Encoderschritte verbunden (#Frage: Wie gerundet?). Die minimalen bzw. maximalen Positionen eines Motors als Encoderschritte werden im ini-File durch Konstanten angegeben: PositionMin, PositionMax.

3.4 Schrittweite

Für die Antriebe DF, DC und TL erfolgt die Eingabe der Schrittweite in Winkelsekunden, für den Antrieb AR in Grad und für den Antrieb CC in Mikrometer. Die zulässigen Wertebereiche findet man für jeden Antrieb als Konstanten im ini-File: MinimalWidth, MaximalWidth

Beispiel (aus einem speziellen ini-File entnommen):

- DC [4 .. 1000],
- DF [2 .. 800],
- TL [2 .. 10.000],
- AR [2 .. 800],
- CC [2 .. 500] .

3.5 Bewegungsgeschwindigkeit

Für die Antriebe DF, DC und TL erfolgt die Eingabe der Bewegungsgeschwindigkeit in Winkelsekunden/Sekunde, für den Antrieb AR in Grad/Sekunde und für den Antrieb CC in Mikrometer/Sekunde. Die zulässigen Wertebereiche findet man als Konstante im ini-File: MaxVelocity für den jeweiligen Antrieb.

Beispiel (aus einem speziellen ini-File entnommen):

- DF [0 .. 8.000],
- DC [0 .. 8.000],
- TL [0 .. 8.000],
- AR [0 .. 8.000],
- CC [0 .. 1.000].

→ Beobachtung

Die eingegebenen Werte für die Geschwindigkeit werden beim Wechsel der Antriebsarten gerundet.

4. Fehler

Siehe im Dokument auch unter #Fehler

1. Im Fahrbetrieb verursacht die Tastaturbenutzung (Pfeiltasten) einen Tastaturpufferüberlauf, da die Taste ständig gedrückt werden muss.

   Lösung: Ereignissteuerung (unter Windows) bei Tastaturdruck

2. Handhabung der Endlagenschalter der Geräte wird softwaremässig nicht ausreichend unterstützt.

5. Änderungswünsche

Siehe im Dokument auch unter #Wunsch

1. Alle Antriebe sollten gleichzeitig bearbeitbar sein, d.h.

   - über getrennte Fenster (je Motor ein Fenster) oder
   - ein Fenster mit der Möglichkeit der gleichzeitigen Bedienung von mehreren (5) Motoren(Achtung: 7 Antriebe bei der Diffraktometrie)

2. Abspeicherung von Zuständen im ini-File:

   ini-File: Feste Vorgabewerte werden bei jedem Programmstart für alle Anfangswerte geladen, womit eine unflexible Arbeitsweise verbunden ist.

   bisherige Arbeitsweise: Editieren im ini-File. um persönliche Einstellungen auf der Grundlage vorheriger Versuchsreihen vorzunehmen. Diese Arbeitsweise ist problematisch und führte in der Vergangenheit zu schwer zu lokalisierenden Fehlern.

   Sinnvoll: Erarbeitete manuelle Einstellungen sollte man in unterschiedlichen Files retten können. Damit kann man mit verschiedenen Ausgangssituationen operieren (Versuchsreihen)

   Kontrollierte Einstellungen der ini-File-Parameter nur im Dialog mit Überprüfung der logischen Konsistenz der Werte sowie der Möglichkeit, die Einstellungen in einem privaten ini-File zu retten (s. o.)

   Achtung: bestimmte gerätespezifische Parameter des ini-Files sollten nach einer gesicherten Einstellung geschützt werden (keine

Notwendigkeit der Änderung im normalen Betrieb). Hierzu zählen: IOADDR, RamAddr, Torque, RemoveLimit, Koeff_1/2/3

damit: Splitten des ini-Files in zwei Dateien:
- geschützte Daten
- änderbare Daten

#Anmerkung: Aufteilung bereits erfolgt.

3. Die Geschwindigkeit V könnte einheitlich auf alle drei Bedienungsarten Einfluss haben.

   (Anm.: Für den Schrittbetrieb reicht die maximale Geschwindigkeit aus - Aussage von Herrn Pfeiffer)

4. Die zugehörige Dialogbox entspricht ergonomisch nicht optimal der angestrebten Funktionalität.
5. Wertebereiche (u. a. für Soll-Position) in Dialogbox als Eingabehilfe?
6. Eingaben für sich bewegende Motoren nicht ignorieren, sondern erst gar nicht ermöglichen (Funktion der Oberfläche)! (Bei Irrtum: erst laufende Bewegung stoppen)
7. Stoppen sich bewegender Motoren derzeit nur über Verlassen der Dialogbox möglich. Stattdessen sollte ein extra Button 'Stop' (des aktuellen Motors) existieren.
8. Bei fehlerhaften Eingaben: Warnton und Frage an den Nutzer, ob auf Grenzwert gesetzt werden soll.

6. Offene Fragen bzw. Maengel im Dokument

(+ geklärt, - noch offen)

Siehe im Dokument auch unter #Frage

1. + Kann ein sich im Direktbetrieb (Schrittbetrieb) bewegender Motor auch gestoppt werden (und wie)? (geklärt: 2.3 f)
2. + Bezieht sich der Schieberegler auf die relative oder absolute Null? (2.3 a)
3. + Kann die relative Null bei laufendem Motor gesetzt werden (eigentlich nicht sinnvoll)? (2.3 b)
4. + Verlassen des Dialogs: Was passiert mit sich bewegenden Motoren (stoppen oder weiterlaufen lassen)? (2.3 f)
5. + Bei DC und TL: Wertebereich der Position kein Vielfaches der Auflösung (Maximalposition nicht erreichbar)!

Antwort: Durch freies Editieren des ini-Files durch den Nutzer kann dieser Fall eintreten, d. h. die Maximalposition kann hier im Schrittbetrieb nicht erreicht werden. Allerdings wird für die Ansteuerung einer Maximalposition eher der Fahrbetrieb verwendet.

6. + Maßeinheit der Schrittweite: Motorschritte ? (nein, s. Kommentierung des ini-Files)
7. + AR-Schrittweite paßt nicht zum AR-Bereich! (Dasselbe Problem wie bei Frage 5.)
8. + AR-Bewegungsgeschwindigkeit: Maßeinheit in Winkelsekunde / Sekunde (nicht: Grad/Sekunde)? (doch: Grad/Sekunde, was aber im speziellen ini-File, das hier als Beispiel dient, im Vergleich mit den anderen Antrieben nicht sinnvoll ist)
9. - Minimale Werte für die Auflösung auch bei Schrittweite (dieselbe wie bei Position) und bei der Geschwindigkeit?
10. + Korrektur ungültiger Eingabewerte (z. B.außerhalb der Bereiche oder nicht entsprechend der Auflösung):Information des Nutzers zu begradigten / geänderten Werten?

    Antwort: Winkelangaben müssen nicht der Auflösung entsprechen (es wird gerundet - wie?)

11. - Wie lange dauern normale Bewegungen bzw. Bewegungen im Ausnahmefall (d. h. ist das Stoppen nötig?)
12. - 1 Winkelsekunde = ?? Motorschritte

    Ist gerätespezifisch und steht noch aus.

13. + Was passiert beim Setzen der relativen Null mit dem Schieberegler?
14. - Informationen zum ini-Files im Text zu unsystematisch
15. - 2.4: Messung der Halbwertsbreite: Beschreibung derzeit noch zu knapp und unklar.
16. - Wie wird (genau) gerundet? (Vergleiche 3.3 Anmerkung 3.)
17. - Woher bekommt die Software den absoluten Nullpunkt?
18. - in 2.3.e: Gliederungskriterium schlecht; statt Hervorheben der Bewegungsarten besser: 1. Soll-Position, 2. Geschwindigkeit, 3. Schrittweite
19. + 2.3.d(relative Null) zu spaet: besser nach 2.3.a (aktuelle Position)

7. Verwandte Dokumente

Pflichtenheft 'Automatische Justage'
Verhaltensspezifikation (Pflichtenheft) 'Motoren'
Verhaltensspezifikation (Pflichtenheft) 'Detektoren'
Ueberblick

8. Aenderungen am Dokument #Frage: Wird dieses Änderungslog überhaupt geführt?

Version 1.1: Aenderungen zur Version 1.01:
- 2.2 erweitert
- 2.3 a, b, d erweitert
- 2.3 f neu
- 3: b, c ,d modifiziert
- 4: 9., 10. neu
- 5: Fragen geklärt und neue aufgenommen

9. Quellen des Dokuments

1. Die Verhaltensspezifikation zur manuellen Justage entstand als erstes Dokument des Projekts im Projektseminar 'Software-Sanierung' im WS 98/99. Grundlage war ein Vortrag des Physikers Herr Pfeiffer in Adlershof zur manuellen Justage. Die Ausgangsversion wurde im Seminar diskutiert und verbessert.
2. Anmerkungen, Januar 99: U. Sacklowski, B. Wolf, Herr Pfeiffer
3. Das vorliegende Pflichtenheft war Gegenstand eines Review in der SE-Vorlesung im SS 99. Es diente als Ausgangspunkt für eine Übung mit dem cte zur Bestimmung von Testfällen, die sich an eine Vorlesung von Herrn J. Wegener von DaimlerChrysler anschloss.
4. Anregungen, Mai 99: J. Wegener (DaimlerChrysler)
5. Die Version 1.01 (14. Juli 1999) wurde von Zoran Budimac (Novi Sad) ins Englische uebersetzt
6. Anmerkungen, August 2000: S. Freund, D. Hepp
7. Im Projektseminar 'SW-Sanierung fuer Neueinsteiger' im WS 2000/01 wurde in Kleingruppen ein weiteres Review des Dokuments vorgenommen. Daraus ergaben sich Anregungen fuer eine grundlegende Revision der Gliederung:
    1. Aufnahme neuer Gliederungspunkte 7., 8., 9., 10.
    2. Neuer Titel: Statt 'Pflichtenheft' (Verwendung bei Neuentwicklung von Software) jetzt 'Verhaltensspezifikation' (Verwendung bei Beschreibung existierender Software).
    3. Standardkommentare im Text: #Frage, #Fehler, #Wunsch
    4. In der VL Software Engineerin im Sommer 2001 erfolgte ein Review. Zahlreiche Änderungsanregungen wurden übernommen. (Kay Schützler)

10. Glossar

arcsec → Winkelmaß

Antrieb → Antriebsart

Motor zur Bewegung der Probe bzw. Krümmung des Kollimators

Antriebsart AR, Azimutale Rotation →

Antrieb zur Drehung des Probentellers. (Verwendung in der Topographie. Bezugsobjekt: Probenteller.)

CC, Kollimator → Kollimator

Antrieb fuer den Kollimator-Krümmer

DC, Beugung Grob (engl.: Diffraction angle coarse

Antrieb zur Grobeinstellung des Beugungswinkels. Der Beugungswinkel liegt in der Ebene Kollimator-Probe-Aufnahmemedieum. (Verwendung in der Topographie. Bezugsobjekt: Probenteller.)

DF, Beugung Fein (engl.: Diffraction angle fine)

Antrieb zur Feineinstellung des Beugungswinkels. Der Beugungswinkel liegt in der Ebene Kollimator-Probe-Aufnahmemedieum. (Verwendung in der Topographie. Bezugsobjekt: Probenteller.)

TL, Tilt

Antrieb zur Verkippung des Probentellers. (Verwendung in der Topographie. Bezugsobjekt: Probenteller.)

Azimutale Rotation -> Antriebsart

Betriebsarten

Direktbetrieb

Bewegung des aktuellen Antriebes mit der eingestellten Fahrgeschwindigkeit von der Ist-Position zur vorgegebenen Soll-Position.

Fahrbetrieb

Bewegung des aktuellen Antriebes mit der eingestellten Fahrgeschwindigkeit solange, wie die Cursortasten (← oder →) der Tastatur oder die Endeelemente des Scrollbars gedrückt gehalten werden.

Schrittbetrieb

Bewegung des aktuellen Antriebes mit der eingestellten Schrittweite von der Ist-Position aus. Die Bewegung wird aktiviert, wenn eine der

Cursortasten der Tastatur (← oder →) betätigt wird bzw. auf dem Scrollbar das linke oder rechte Endeelement angeklickt wird. Die Fahrgeschwindigkeit entspricht der maximalen.

Beugung Fein → Antriebsart

Beugung Grob → Antriebsart

Bragg-Reflex

maximale konstruktive Interferenz der an parallelen Netzebenen eines Einkristalls reflektierten monochromatischen Röntgenstrahlung. Es gilt die Bedingung 2dsin(Theta)=n(Lambda), wobei d der Abstand der parallelen Netzebenen, Theta der Einfallswinkel der Strahlung auf die Netzebene, Lambda die Wellenlänge des Röntgenlichts und n eine beliebige natürliche Zahl ist. Da jeder Kristall über eine sehr große Zahl von Netzebenen verfügt, gibt es mehrere solcher Reflexe. Aus den gemessenen Winkeln läßt sich die Kristallstruktur erschließen. Dieser Wert hilft bei der Justierung der Probe.

Direktbetrieb → Betriebsarten

Fahrbetrieb → Betriebsarten

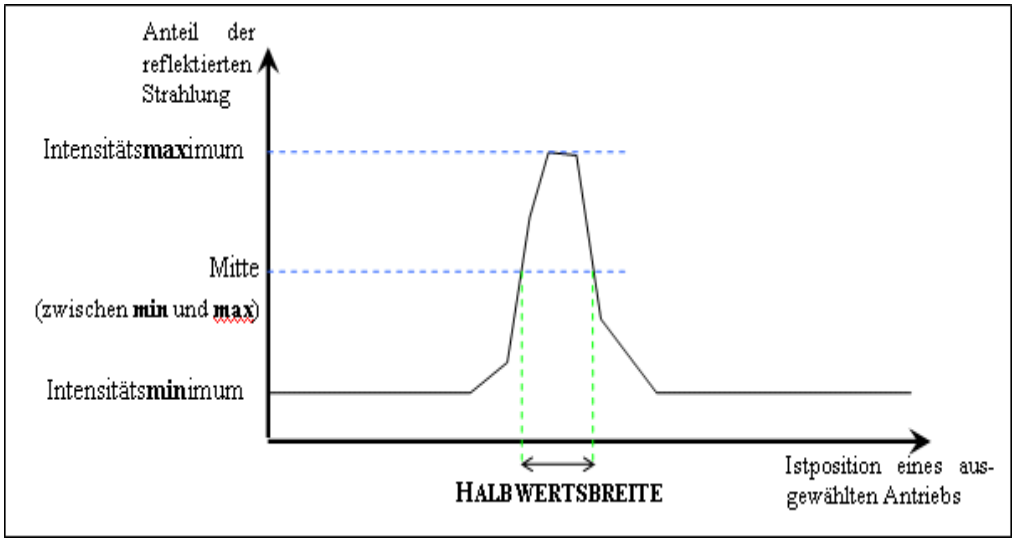
Feinjustage

abschließender Vorgang der manuellen Justage, in dem die Feineinstellungen über Beubung Fein, Tilt und Kollimator vorgenommen werden

Grad → Winkelmaß

Halbwertsbreite

Differenz der Abszissen-Werte (x-Achse, in der Praxis meist Winkel oder Entfernungen) links und rechts eines Ordinatenmaximums (y-Achse, üblicherweise Intensität), bei denen der gemessene Wert nur noch die Hälfte des Maximums beträgt. Gelegentlich findet man den Faktor 1/(Wurzel 2) anstelle von 1/2. Dies ist dann der Fall, wenn der Meßwert eine Amplitude darstellt, deren Quadrat proportional zur Intensität ist.



Kollimator → Antriebsart

eigentlich eine optische Vorrichtung, um in einen Strahlengang eine Meßskala so einzufügen, daß sie sich für den Beobachter im Unendlichen befindet. Hier wird ein Kollimatorkristall dazu verwendet, verschiedene Wellenlängen in unterschiedliche Richtungen zu reflektieren, um so aus dem ursprünglich kontinuierlichen Röntgenspektrum eine definierte Wellenlänge für die Messung herauszugreifen. Durch geschickte Kombination mehrerer solcher Kollimatoren können Unebenheiten und Krümmungen des zu untersuchenden Kristalls ausgeglichen werden. Darüber hinaus bewirkt der Kollimator eine Verbreiterung des Röntgenstrahls.

Meßprobe

Kristallprobe, die einer Topographiemessung unterzogen wird

Minute, min → Winkelmass

Motor → Antrieb

Peak

Intensitätsmaximum der untersuchten Strahlung

Probenteller

der Teller, auf den die Probe gelegt wird

Rocking-Kurve

Intensitätsverteilung der untersuchten Strahlung in einer Dimension. Die Halbwertsbreite dieser Kurve dient als Qualitätskriterium bei der Justierung der Probe.

Schrittbetrieb → Betriebsarten

Sekunde, sec → Winkelmaß

Tilt → Antriebsart

Winkelmaß

Grad

Einheit des ebenen Winkels. Ein Grad ist der 90stel Teil des rechten Winkels.

Winkelminute, Minute, min
60stel Teil eines Grades, Zeichen:'

Winkelsekunde, Sekunde, sec
60stel Teil einer Minute, Zeichen:"

Arkus, Arcus, arc
Bogenmaß eines Winkels

arcsec
Wird in dem Projekt gleichbedeutend mit der Winkel-Sekunde benutzt. Der Grund ist, eine Verwechslung mit der Zeit-Sekunde auszuschließen.

LITERATURVERZEICHNIS

[1] Bothe, Klaus – Reverse Engineering: the Challenge of Large-Scale Real-World Educational Projects, CSEE&T 2001, 14th Conference on Software Engineering Education and Training, Charlotte, USA, 02.2001

[2] Bowen, Jonathan - Formal Specification and Documentation using Z: A Case StudyApproach; International Thomson Computer Press, International Thomson Publishing, 1996; www.zuser.org/zbook/; revised 2003

[3] Spivey, Mike - The Z Notation: A Reference Manual, 2nd edition; Prentice Hall International Series in Computer Science, 1992; spivey.oriel.ox.ac.uk/mike/zrm/

[4] Bothe, Klaus - Verhaltensspezifikation (Pflichtenheft) XCTL-Steuerprogramm - Funktion: Probe und Kollimator manuell justieren; Dokumentversion: 2.2 (4.7.2001)
www2.informatik.hu-berlin.de/swt/projekt98/entwicklerdoku/entwicklertabelle/manujustage/Manuelle_Justage.v2.2.html

[5] U. Sacklowski - Verhaltensspezifikation XCTL-Steuerprogramm Funktion: 0-dimensionale Detektoren Teilfunktion: Zählerfenster; Dokumentversion: 2.04 (15.06.2006)
http://www2.informatik.hu-berlin.de/swt/projekt98/entwicklerdoku/entwicklertabelle/detektornutzung/0-dim-detektoren/index.html

[6] J-M.Bruel, A.Benzekri, Y.Raynand „Z and the Specification of Real-Time Systems"; Proceedings, "7th international Conference on: Putting into practice methods and tools for information system design", Nantes (France), October 95, IUT de Nantes, H. Habrias (editor)

[7] C.J. Fridge "Specification and Verification of Real Time Behaviour Using Z and RTL"

[8] A. Coombes, John McDermid "Specifying Temporal Requirements for Distributed Real-Time Systems in Z"
http://www.cs.york.ac.uk/ftpdir/reports/YCS-92-176.pdf

[9] J.Lange “Formale Spezifikation von „Manuelle Justage (Alt)“ ein Teilsystem des XCTL – Steuerprogramms“ Studienarbeit

[10] Davis, Jim; Woodcock, Jim - Using Z: Specification, Refinement and Proof; Prentice Hall International Series in Computer Science, 1996; www.usingz.com

[11] Jacky, Jonathan - The Way of Z: Practical Programming with Formal Methods, Cambridge University Press, 1997

[12] Currie, Ed - The Essence of Z; Prentice-Hall, 1999