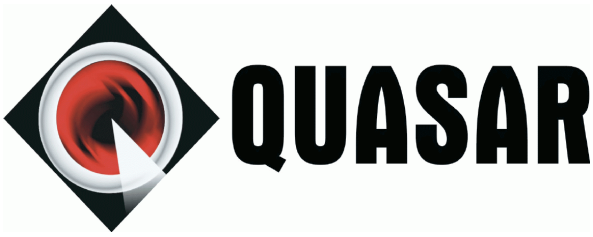
Fraunhofer Institut
Experimentelles
Software Engineering

Modellierung eines Türsteuergerätes mit SCR

Eine Fallstudie im QUASAR-Projekt



Autoren:
Christian Denger
Jörg Dörr
Erik Kamsties

IESE-Bericht Nr. 064.01/D
Version 1.0
22. Oktober 2001

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer Gesellschaft.
Das Institut überträgt innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach
Sauerwiesen 6
67661 Kaiserslautern

Abstract

Die Modellierung des Verhaltens von (eingebetteten) Systemen mit Hilfe von Modellierungssprachen ist ein wichtiger Beitrag, um die Anforderungen an ein System besser zu verstehen, besser zu kommunizieren und fundierte Aussagen über die Qualität von Anforderungen machen zu können. In Forschung und Praxis finden verschiedene Modellierungssprachen Anwendung. Während bei der Entwicklung von eingebetteten Systemen in der Praxis häufig die Sprache Statecharts (unterstützt durch Statemate- oder Rhapsody-Tool) eingesetzt wird, untersuchen wir im folgenden die Sprache SCR (Software Cost Reduction) vom U.S. Marineforschungsinstitut (U.S. Naval Research Lab), die sich im Vergleich durch eine sehr einfache Semantik und hohe Lesbarkeit und damit auch Verständlichkeit auszeichnet.

Dieser Bericht dokumentiert die Modellierung eines Türsteuergerätes (TSG) mit Hilfe von SCR. Das TSG ist ein Steuergerät, das in modernen Fahrzeugen eingesetzt wird, um die Ansteuerung von Außenspiegeln, Fenstern und Sitzen zu kontrollieren. Ausgangspunkt war eine fiktive textuelle Beschreibung eines TSG von DaimlerChrysler.

Der Bericht gibt eine kurze Einführung in die Sprache SCR und die Vorgehensweise bei der Modellierung. Anschließend wird das resultierende Modell im Einzelnen beschrieben. Vorallem die Realisierung von besonderen Anforderungen wie Prioritätenregelungen und die Modellierung der Umgebung sowie Erfahrungen hinsichtlich der Anwendbarkeit von SCR in der Beispieldomäne werden diskutiert. Darüberhinaus wurden während der Modellierung durch den Einsatz von SCR Fehler und Inkonsistenzen in der Beispiel-Spezifikation aufgedeckt, die ebenfalls beschrieben werden.

Schlagworte: Requirements, Embedded Systems, Software Cost Reduction (SCR), Modellierung, Quasar

Acknowledgement: This work was done as part of the QUASAR project, which is funded by the German BMBF under the grant VFG0004A.

Inhaltsverzeichnis

1 Einleitung

Im folgenden Bericht wird die Modellierung eines Türsteuergerätes (TSG), ausgehend von einer Beispielspezifikation [TSG], mit der Modellierungssprache SCR (Software Cost Reduction) beschrieben. Ein solches TSG ist ein eingebettetes System. Es ermöglicht es dem Fahrer des Fahrzeuges z.B. die Position seines Sitzes und der Aussenspiegel nach seinen Wünschen einzustellen.

Ziel der Modellierung war es, eine Ausgangsbasis zu bieten, um die Eignung von SCR-Modellen als Teil des Systemspezifiaktionsdokuments zu bewerten (vgl. Kamsties, von Knethen, Paech: Structure of QUASAR requirements documents).

Das in diesem Bericht dokumentierte Modell repräsentiert lediglich einen Ausschnitt der Gesamtfunktionalität eines TSG, nämlich die Funktionalität der Sitzeinstellung. Gegebenenfalls werden weitere Funktionalitäten zu einem späteren Zeitpunkt ergänzt.

In Kapitel 2 wird zunächst eine kurze Einführung in die Modellierungssprache SCR gegeben. Es werden nur die groben Konzepte der Sprache erläutert. Eine detaillierte Beschreibung von SCR ist im Anhang zu finden (Anhang A: Introduction to SCR).

Kapitel 3 erläutert die im Modell verwendeten monitorierten und kontrollierten Variablen, die die Schnittstellen des TSGs modellieren. Dabei werden die Variablen wegen ihrer Vielzahl in logischen Gruppen zusammengefaßt und diese Gruppen summarisch beschrieben, da sich die Variablen häufig nur im Namen unterscheiden, nicht in ihrer grundsätzlichen Funktionalität. Weiterhin werden die verschiedenen Modi (Zustände) des Modells und Terme (Makros) des Modells erläutert.

In Kapitel 4 werden Besonderheiten des Modells aufgezeigt. Dazu zählen die Realisierung von Constraints und Prioritäten. Weiter wird die Modellierung der Umgebung des TSGs und die Modellierung der Schnittstellen zur Umgebung des TSGs näher beschrieben.

Erfahrungen, die während der Modellierung gesammelt wurden, sowie wünschenswerte Erweiterungen des SCR-Toolsets werden in Kapitel 5 zusammengefasst.

Entwurfsentscheidungen die bei der Erstellung des Modells getroffen wurden werden in Kapitel 6 beschrieben. Weiter werden Besonderheiten des Modells

erläutert, wie z.B. nichtrealisierte Anforderungen oder bekannte Abweichungen des Modells von den informellen TSG-Anforderungen.

In Kapitel 7 werden die während der Modellierung erstellten Event-Scripts für den SCR-Simulator kurz erläutert.

Der Begriff "Modell" wird in diesem Bericht sowohl für die entstandene tabellarische Beschreibung des TSG in SCR verwendet, wie auch für einen Ausführungszustand der tabellarischen Beschreibung. Beispiel: *Dem SCR Toolset fehlt eine Funktion, um nach der Verwendung einer bestimmten Variablen oder eines Terms in einem Modell (=tabellarische Beschreibung) zu suchen. Das Modell (= Ausführungszustand) wechselt vom Zustand "Sitz_bewegen_1" nach "Sitz_bewegen_2", wenn ...*

2 Die Modellierungssprache SCR

Die Software Cost Reduction (SCR) Spezifikationstechnik wurde für das Gebiet der reaktiven Systeme entwickelt. SCR ist heute eine formale Beschreibungstechnik für System- und Software-Anforderungen von reaktiven Systemen. Die Technik wurde von Parnas et al. vor zwei Jahrzehnten am Naval Research Lab als Teil des Software Cost Reduction Projektes entwickelt, um das operationale Flugprogramm für das Navy A-7 Flugzeug kurz und eindeutig zu spezifizieren. Heitmeyer et al. verstärkten die formalen Grundlagen SCR's, indem sie eine formale Semantik hinzufügten. Weiterhin entwickelten sie die passende Werkzeugunterstützung die als SCR toolset bekannt ist.

Ein reaktives System wird in SCR als endlicher Zustandsautomat dargestellt. Die Zustände werden Modi genannt und Übergänge werden durch Events gestartet. Das geforderte Systemverhalten wird durch einige wenige Konstrukte beschrieben: monitorierte und kontrollierte Variablen, Modes, Modeklassen, Terme, Conditions und Events. Eine monitorierte Variable stellt einen Aspekt der Umgebung dar, der das Systemverhalten beeinflußt, kontrollierte Variable stellen einen Aspekt der Umgebung dar, der vom System gesteuert wird. Ein Modus ist eine Teilmenge des Zustandsraumes des Systems. Eine Modeklasse ist eine Komposition von Modi derart, daß die Modi einer Modeklasse gegenseitig exklusiv und vollständig sind, d.h. eine Modeklasse ist ein Zustandautomat. Ein Term ist ein Makro für einen komplizierten Ausdruck von monitorierten, kontrollierten Variablen, Modi oder anderen Termen. Eine Condition ist ein Prädikat über monitorierten Variablen, Modi oder Bezeichnungen. Ein Event tritt auf, wenn eine monitorierte oder kontrollierte Variable, ein Term oder eine Modeklasse seinen/ihren Wert ändert. Modeklassen werden durch Mode Transition Tables definiert und Terme und kontrollierte Variablen werden durch Eventtables oder Conditiontables definiert. Ein Abhängigkeitsdiagramm (Dependency Graph) zeigt die "benutzt"-Abhängigkeiten zwischen monitorierten Variablen, Termen, Modeklassen und kontrollierten Variablen und ermöglicht somit Zyklen einfacher zu erkennen.

Tabelle 1 auf Seite 4 zeigt eine Eventtabelle. Diese Tabelle beschreibt die Änderungen der kontrollierten Variable 'cSmot_h' in Abhängigkeit vom Eintritt verschiedener Events z.B. @T(tHinten_bewegen = TRUE) AND (mSitz_h = heben)'.

Weitere Erläuterungen zu SCR anhand eines Beispieles und ein Prozeß wie die SCR Spezifiationstechnik angewendet werden kann sind im Anhang A zu finden.

Tabelle 1: Beispiel einer Event-Table

Name		Mode Class	
cSmot_h		Sitzeinstellung	
Modes	**Events**		
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = heben)')	(@T(tHinten_bewegen = TRUE) AND (mSitz_h=senken)')	@T(tHor_bewegen = TRUE OR tSchalung_bewegen = TRUE OR tVorne_bewegen = TRUE OR tWinkel_bewegen = TRUE)
cSmot_h' =	heben	senken	ruhen

3 Der Aufbau des Modells

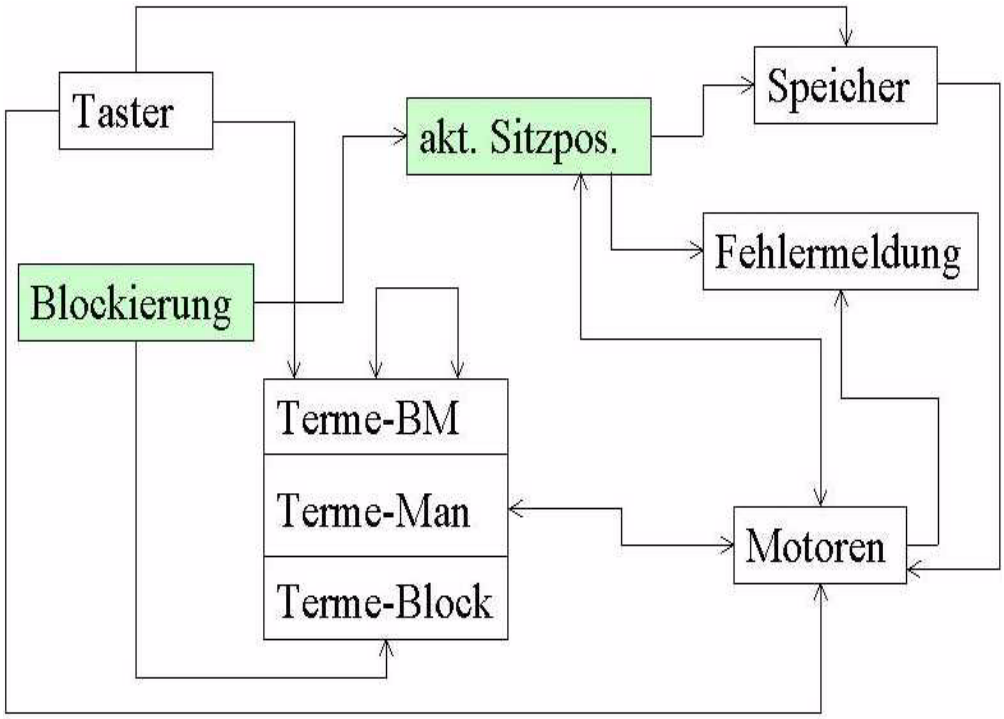
Kapitel 3 gibt erst einen Überblick über den Aufbau des Modells, und beschreibt dann die im Modell verwendeten monitorierten und kontrollierten Variablen. Erfüllen Variablen gleiche Aufgaben so werden sie in einer Gruppe zusammengefaßt und durch diese Gruppe summarisch beschrieben. Desweiteren werden noch die Modi in denen sich das Modell befinden kann und einige grundlegende Terme des Modells erläutert, die nicht zur Realisierung einer besonderen Funktionalität dienen, sondern zur besseren Verständlichkeit des Modells eingeführt wurden. Terme, zur Realisierung einer besonderen Funktionalität dienen, werden im folgenden Kapitel beschrieben. Am Ende dieses Kapitels wird ein beispielhafter Auszug aus dem Modell näher erläutert.

Eine Besonderheit des Modells ist die Beschreibung eines Teiles der Funksendereinheit. Diese Einheit empfängt das Funksendersignal des Schlüssels und gibt es über den CAN-Bus an das TSG weiter (siehe Abschnitt 3.3.2).

Weiterhin ist zu beachten, daß die Verfolgbarkeit von den monitorierten, kontrollierten Variablen und den Termen des Modells zu dem Anforderungsdokument [TSG] in dem Modell selbst (Beschreibung der Variablen, siehe Anhang C) dokumentiert ist.

3.1 Übersicht über den Aufbau

Die folgende Grafik illustriert den groben Aufbau des Modells. In dieser Grafik sind Gruppen von monitorierten bzw. kontrollierten Variablen sowie Termen zusammengefaßt und als Rechtecke dargestellt (z.B. Taster). Ein Pfeil bedeutet, daß eine Gruppe an der Spitze des Pfeiles von Termen/ Variablen am Ende des Pfeiles direkt abhängt (z.B. wenn eine Variable am Ende des Pfeiles in der Eventtabelle einer Variablen an der Spitze des Pfeiles vorkommt). Dieses Bild gibt nicht das vollständige Modell wieder, es konzentriert sich auf die wesentlichen Elemente.



Die unterlegten Rechtecke sind Gruppen, die zum Umgebungsmodell gehören, alle anderen gehören zum Systemmodell (d.h. zum eigentlichen TSG). Es wurde auch die Umgebung modelliert, um das Systemmodell vollständig simulieren zu können.

Die Abbildung zeigt beispielsweise, daß die aktuelle Sitzposition sowohl von der Ansteuerung der Motoren, als auch von dem Umstand, ob ein Objekt aus der Umgebung eine Blockierung auslöst oder nicht, abhängt. Tritt eine Blockierung auf, so wird eine Fehlermeldung generiert. Diese hängt sowohl von der Ansteuerung der Motoren als auch von der aktuellen Sitzposition ab. Dies begründet sich darin, daß eine Blockierung dann erkannt wird, wenn ein Motor angesteuert wird, den Sitz also bewegen soll, sich aber die aktuelle Sitzposition aber nicht verändert. Im folgenden werden die einzelnen Gruppen von monitorierten bzw. kontrollierten Variablen sowie die Termgruppen näher erläutert. Weiterhin wird am Ende des Kapitels der direkte Zusammenhang von Tastern und Motoren sowie deren indirekten Zusammenhang über die Terme der manuellen Sitzeinstellung an einem Ausschnitt aus dem Modell erläutert.

3.2 Monitorierte Variablen

Die monitorierten Variablen beschreiben Werte, die dem TSG aus der Umgebung (z.B über Sensoren oder den CAN-Bus) zur Verfügung gestellt werden. Die Variablen werden in Gruppen eingeteilt und anhand dieser Gruppe beschrieben.

Folgende monitorierten Variablen wurden im Modell verwendet:

Abschnitt 3.2.1, "Realisierung der Umgebung: Blockierung":
mBlockiert_h, mBlockiert_hor, mBlockiert_s, mBlockiert_v, mBlockiert_w

Abschnitt 3.2.2, "Simulation der Taster zur manuellen Sitzeinstellung":
mSitz_h, mSitz_hor, mSitz_s, mSitz_v, mSitz_w

Abschnitt 3.2.3, "Simulation der Taster zur automatischen Sitzeinstellung":
mFunksender_Pos1, mFunksender_Pos2, mMgmt_1, mMgmt_2, mMgmt_set

Abschnitt 3.2.4, "Andere Fahrzeugparameter":
mBatt, mIn_Fzg_v, mT_offen

3.2.1 Realisierung der Umgebung: Blockierung

Die Blockierung eines Sitzes und der entsprechenden Motoren, z.B. durch einen großen/schweren Gegenstand auf dem Sitz, muß über monitorierte Variablen modelliert werden, da der Sitz durch die Umgebung blockiert wird. Da das TSG für die Ansteuerung von 5 Sitzmotoren verantwortlich ist, werden hierzu 5 monitorierte Variablen benötigt (**mBlockiert_h**, **mBlockiert_hor**, **mBlockiert_s**, **mBlockiert_v**, **mBlockiert_w**), die die Blockierung des jeweiligen Motors modellieren.

3.2.2 Simulation der Taster zur manuellen Sitzeinstellung

Zur Ansteuerung der Sitzmotoren kann der Benutzer des TSGs 5 Taster (mit 3 möglichen Positionen) verwenden, die jeweils einem Motor für eine Bewegung in eine Richtung zugeordnet sind. Diese Taster werden mittels der 5 Variablen **mSitz_h** (Fahrersitz hinten heben bzw. senken), **mSitz_hor** (Sitz horizontal bewegen d.h. nach vorne bzw. nach hinten), **mSitz_s** (Schalung des Sitzes enger bzw. weiter) **mSitz_v** (Sitz vorne heben bzw. senken) und **mSitz_w** (Winkel der Lehne steiler oder flacher einstellen) modelliert.

3.2.3 Simulation der Taster zur automatischen Sitzeinstellung

Hier kann man noch zwei Untergruppen identifizieren. Der Benutzer kann entweder die automatische Einstellung des Sitzes mit einem Funksender initiieren (über Variablen **mFunksender_Pos1** bzw. **mFunksender_Pos2**) oder durch die Managementtaster (**mMgmt_1** bzw. **mMgmt_2**).

Das TSG erlaubt es 4 verschiedene Sitzpositionen im Speicher des TSG abzulegen. Das Modell simuliert aus Vereinfachungsgründen lediglich 2 solche Speicherpositionen. Man benötigt zum Abrufen einer Speicherposition jeweils eine eigene Variable (zu erkennen an der Nummern der Variablen 1 bzw. 2). Wäre eine Parametrisierung in SCR möglich, so hätte man an dieser Stelle davon Gebrauch machen können (siehe Kritik in Abschnitt 5.2).

Mittels der Managementtaster kann eine Sitzposition auch im Speicher abgelegt werden. Dazu ist eine weitere Taste vorgesehen (modelliert über die Variable **mMgmt_set**), die in Kombination mit einer Managementtaste (mMgmt_1 oder mMgmt_2) gedrückt werden muß, um festzulegen, an welcher Speicherposition die Werte der Sitzpositionen abgelegt werden.

3.2.4 Andere Fahrzeugparameter

Hierzu zählen Voraussetzungen, die für die Einstellung des Sitzes gelten müssen, damit eine Bewegung der Motoren angestoßen werden kann. Diese Voraussetzungen beinhalten, daß die Batteriespannung über einem festgelegten Schwellwert liegen muß (modelliert mit der Variable **mBatt**). Weiter darf die Fahrzeuggeschwindigkeit je nach Modus (siehe Kapitel 2.3) eine bestimmte Geschwindigkeit nicht überschreiten (**mIn_Fzg_v**), damit die Sitzposition noch verändert werden darf. Zur Ansteuerung der Motoren mittels der Taster muß die Fahrzeugtür geöffnet sein. Dies ist mit Hilfe der Variable **mT_offen** modelliert.

3.3 Kontrollierte Variablen

Die kontrollierten Variablen sind Größen, die vom System manipuliert werden. Sie können z.B für Aktuatoren stehen, aber auch für Speicherpositionen, da Speicher in SCR nicht explizit modelliert werden können.

Im folgenden werden die kontrollierten Variablen des Modells nach Gruppen geordnet erläutert.

Im Modell wurden folgende kontrollierte Variablen verwendet:

Abschnitt 3.3.1, "Fehlermeldungen":
cFehler_sitz_blockiert, cOut_b_low_sitz

Abschnitt 3.3.2, "Speicherabruf durch Funksender":
cIn_M_pos_1, cIn_M_pos_2

Abschnitt 3.3.3, "Sitzansteuerung":
cSmot_h, cSmot_hor, cSmot_v, cSmot_w, cSmot_s

Abschnitt 3.3.4, "Speicherpositionen":
cSpeicher_Pos1_h, cSpeicher_Pos1_hor, cSpeicher_Pos1_v,
cSpeicher_Pos1_w, cSpeicher_Pos1_s,
cSpeicher_Pos2_h, cSpeicher_Pos2_hor, cSpeicher_Pos2_v,
cSpeicher_Pos2_w, cSpeicher_Pos2_s

Abschnitt 3.3.5, "Aktuelle Sitzposition":
cSpos_h, cSpos_hor, cSpos_v, cSpos_w, cSpos_s

Abschnitt 3.3.6, "Werte gespeichert":
cWerte_gespeichert_1, cWerte_gespeichert_2

Abschnitt 3.3.7, "Nachrichten an das Beifahrer-TSG":
cZu_Beifahrertsg_M_pos_mgmt_1, cZu_Beifahrertsg_M_pos_mgmt_2,
cZu_Beifahrertsg_M_save_1, cZu_Beifahrertsg_M_save_2

3.3.1 Fehlermeldungen

Die kontrollierte Variable **cFehler_sitz_blockiert** modelliert einen Speicherplatz im internen Fehlerspeicher des TSG. Sie wird gesetzt sobald eine Blockierung des dem TSG zugehörigen Sitzes festgestellt wird.

Die kontrollierte Variable **cOut_b_low_sitz** modelliert die Can-Botschaft, die vom TSG abgesendet wird wenn erkannt wird, daß die Batteriespannung zu niedrig ist um die Sitzmotoren zu betreiben.

3.3.2 Speicherabruf durch Funksender

Die Variablen **cIn_M_pos_1** und **cIn_M_pos_2** modellieren den Abruf der zugehörigen Speicherposition von der Funksendereinheit über den CAN-Bus.

3.3.3 Sitzansteuerung

Die Variablen **cSmot_h**, **cSmot_hor**, **cSmot_v**, **cSmot_w**, **cSmot_s** dienen der Ansteuerung der verschiedenen Sitzmotoren. Sie werden genutzt um die Sitzmotoren anzusteuern um den Sitz zu heben/senken, verengen/erweitern bzw. um ihn nach vorne/hinten zu schieben.

3.3.4 Speicherpositionen

Die Variablen **cSpeicher_Pos1_h**, **cSpeicher_Pos1_hor**, **cSpeicher_Pos1_v**, **cSpeicher_Pos1_w** und **cSpeicher_Pos1_s** modellieren jeweils einen Speicherplatz im internen Speicher des TSG. In Ihnen werden die zugehörigen Sitzpositionen des Sitzes für den ersten Benutzer im Benutzermanagement eingestellt. Dies gilt analog für den zweiten Benutzer durch die Variablen **cSpeicher_Pos2_h**, **cSpeicher_Pos2_hor**, **cSpeicher_Pos2_v**, **cSpeicher_Pos2_w** und **cSpeicher_Pos2_s**.

3.3.5 Aktuelle Sitzposition

Die Variablen **cSpos_h**, **cSpos_hor**, **cSpos_v**, **cSpos_w** und **cSpos_s** geben die aktuelle Position des Sitzes an. Sie entsprechen den Werten der Sensoren zur Feststellung der Sitzposition. Eigentlich ist es sinnvoll Variablen, die Sensoren darstellen, als monitorierte Variablen zu deklarieren. Da SCR aber keine Möglichkeit bietet Werte von aussen automatisch zu setzen (z.B. über einen Stub, der die Umgebung simuliert) sind die Sensorvariablen kontrollierte Variablen, und die Veränderung der Sitzposition durch die Motoren Teile des Modells. D.h. Umgebungsmodell und Systemmodell können nicht trennen beschrieben werden.

3.3.6 Werte gespeichert

Die Variablen **cWerte_gespeichert_1** und **cWerte_gespeichert_2** sind Hilfsvariablen, die angeben, ob seit Einbau des TSGs in das Fahrzeug jemals Werte in der Speicherposition 1 bzw. 2 abgespeichert wurden. Ist dies nicht der Fall, darf das TSG nicht versuchen die dadurch undefinierten Positionen einzustellen. (In SCR wären die Positionen natürlich definiert, da dort ja Initialwerte in den Variablen stehen, dies kann man aber in der Realität nicht voraussetzen.) In der Spezifikation des TSG gibt es vier Speicherpositionen. Zur Vereinfachung wurden aber nur zwei Positionen modelliert.

3.3.7 Nachrichten an das Beifahrer-TSG

Die Variablen **cZu_Beifahrertsg_M_pos_mgmt_1**, **cZu_Beifahrertsg_M_pos_mgmt_2**, **cZu_Beifahrertsg_M_save_1** und **cZu_Beifahrertsg_M_save_2** sind Nachrichten, die an das Beifahrer-TSG gesendet werden.

cZu_Beifahrertsg_M_pos_mgmt_1 und **cZu_Beifahrertsg_M_pos_mgmt_2** dienen dazu dem Beifahrer-TSG mitzuteilen, daß die Speicherposition 1 bzw. 2 eingestellt werden soll. Dies ist beim Beifahrer-TSG nur für die Aussenspiegelein-

stellung wichtig, die Sitzposition des Beifahrersitzes wird nicht mit abgespeichert. Die Werte für die Aussenspiegeleinstellung müssen dann aus dem internen Speicher ausgelesen und eingestellt werden (Beachte: Aussenspiegeleinstellung wurde nicht modelliert).

cZu_Beifahrertsg_M_save_1 und **cZu_Beifahrertsg_M_save_2** veranlassen das Beifahrer-TSG dazu die aktuelle Position der Aussenspiegel in den internen Speicher abzulegen.

3.4 Modeklassen

Die Modi (d.h. möglichen Zustände) des TSG sind in die zwei orthogonalen Modeklassen **Sitzeinstellung** und **Taktgeber** aufgeteilt. Im folgenden werden die Modeklassen erläutert.

Im Modell wurden folgende Modi verwendet:

Abschnitt 3.4.1, "Modi der Modeklasse Sitzeinstellung":
Ruhen, Tuer_offen, Bm_Funksender,Sitz_bewegen_1, Sitz_bewegen_2, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken

Abschnitt 3.4.2, "Modi der Modeklasse Taktgeber":
Move, Elapsed

3.4.1 Modi der Modeklasse Sitzeinstellung

In den Modi **Ruhen** und **Tuer_offen** sind noch keine Anforderungen an die Sitzeinstellungen gestellt, die Sitze sollen sich also nicht bewegen, es läuft kein Sitzmotor. In Ruhen, befindet sich das Auto wenn die Fahrertür geschlossen ist, in Tuer_offen, wenn sie geöffnet ist.

Die Sitzeinstellung wechselt in den Mode **BM_Funksender**, wenn ein Signal durch den Funksender eingeht (c_In_M_pos_1 bzw. c_In_M_pos_2) und die Bewegung möglich ist, d.h. daß z.B. die Batteriespannung ausreichend ist. Die Sitzeinstellung verläßt diesen Mode, wenn die Sitzpositionen erreicht, Blockierungen aufgetreten sind bzw. Ereignisse mit höherer Priorität eingehen.

In **Sitz_bewegen_1** bzw. **Sitz_bewegen_2** geht die Sitzeinstellung, wenn der Benutzer die Taster zur manuellen Sitzeinstellung betätigt (siehe 2.1.3). Betätigt er einen Taster, so wechselt die Sitzeinstellung in Sitz_bewegen_1 und ein Sitzmotor läuft. Betätigt er noch einen Zweiten, so wechselt die Sitzeinstellung von Sitz_bewegen_1 nach Sitz_bewegen_2 und ein zweiter Sitzmotor läuft. Betätigt er mehr als zwei Taster, werden diese Tastendrücke ignoriert. Ein direkter Übergang z.B. von Ruhen nach Sitz_bewegen_2 ist nicht möglich, da in SCR keine

zwei Events parallel auftreten können, die von zwei verschiedenen monitorierten Variablen ausgelöst werden. Läßt er den/die Taster los, so wechselt die Sitzeinstellung zuerst in Sitz_bewegen_1 und dann, falls er beide losläßt, nach Ruhen bzw. Tuer_offen.

In die Modi **Sitz_bewegen_entspannen** und **Sitz_bewegen_einschraenken** geht die Sitzeinstellung, wenn eine Änderung der Sitzposition durch das Drükken eines Benutzermanagementtasters gefordert wird. Die Aufteilung in die Modi Sitz_bewegen_entspannen und Sitz_bewegen_einschraenken resultiert aus der in der TSG-Spezifikation geforderten Prioritätenregelung (siehe Abschnitt 4.1.3). Wird ein Benutzermanagementtaster gedrückt und gibt es eine Bewegung, die zur Entspannung der Sitzposition führt, so wechselt die Sitzeinstellung in den Mode Sitz_bewegen_entspannen. Gibt es keine entspannenden Bewegungen bzw. sind alle Bewegungen, die zur Entspannung führen ausgeführt, wechselt die Sitzeinstellung in Sitz_bewegen_einschraenken, sofern es Bewegungen gibt, die zur Einschränkung führen. Gibt es keine einschränkenden Bewegungen bzw. sind diese abgeschlossen, so wechselt die Sitzeinstellung nach Ruhen bzw. Tuer_offen. Während die Sitzeinstellung in Sitz_bewegen_entspannen ist, werden die Sitzmotoren, die zu einer Entspannung der Sitzposition führen, entsprechend einer Prioritätenregelung (siehe Abschnitt 4.1.3) angesteuert. Ist die Sitzeinstellung in Sitz_bewegen_einschraenken, so werden die Sitzmotoren, die zu einer Einschränkung der Sitzposition führen, entsprechend dieser Prioritätenregelung angesteuert.

3.4.2 Modi der Modeklasse Taktgeber

Die Modeklasse Taktgeber mit ihren Modi Move und Elapsed ist hinzugefügt worden um periodische Änderungen zu realisieren. Diese werden benötigt um die Veränderung der Sitzposition zu realisieren. Solange ein Taster zur manuellen Veränderung gedrückt ist, bzw. die gewünschte Sitzposition nach Betätigung eines Benutzermanagementtasters oder des Funksenders noch nicht erreicht ist, muß die Sitzposition periodisch geändert werden. Eine Änderung findet immer dann statt, wenn der Taktgeber in die Mode Elapsed springt. Dies geschieht alle MOVING_TIME (eine der SCR-Konstanten) Einheiten.

3.5 Terme

In diesem Abschnitt werden alle Terme des Modells erläutert, die zur Vereinfachung der Lesbarkeit des Modells eingeführt wurden. Die Terme, die zur Realisierung einer bestimmten Funktionalität des TSGs erzeugt wurden, werden in Kapitel 4 näher beschrieben.

Es werden hier folgende Terme beschrieben:

Abschnitt 3.5.1, "Einnehmen von Speicherpositionen":
tAktuelle_Speicher_Position, tSpeicher_Position1_abgleich, tSpeicher_Position2_abgleich

Abschnitt 3.5.2, "Manuelle Bewegung von Motoren":
tHor_bewegen, tSchalung_bewegen, tVorne_bewegen, tHinten_bewegen, tWinkel_bewegen

3.5.1 Einnehmen von Speicherpositionen

Wenn der Benutzer eine automatische Einstellung des Sitzes initiiert (z.B. über den Funksender mFunksender_Posx oder die Managementtaster mMgmt_x wobei x = 1 oder 2), dann sollen die Sitzmotoren nur aktiviert werden, wenn sich die aktuelle Position des Sitzes von der Speicherposition unterscheidet. Um zu speichern welcher Funksender (mFunksender_Pos1 oder Pos2) bzw. welcher Managementtaster (mMgmt_1 oder mMgmt_2) die Sitzeinstellung ausgelöst hat wurde der Term **tAktuelle_Speicher_Position** eingeführt, der den Wert der Speicherposition enthält (1 oder 2), die eingenommen werden soll, ansonsten hat der Term den Wert 0.

Weiter werden zwei Terme benötigt, die True liefern wenn die aktuelle Sitzposition mit den Werten der Speicherposition 1 (**tSpeicher_Position1_abgleich**) bzw. der Speicherposition 2 (**tSpeicher_Position2_abgleich**) übereinstimmen. Es wird die aktuelle Sitzposition mit der Speicherposition verglichen, die im Term tAktuelle_Speicher_Position angegeben ist. Liefert der entsprechende Term True werden die Motoren nicht in Bewegung gesetzt, liefert er False werden die Motoren solange bewegt, bis der Term True wird.

3.5.2 Manuelle Bewegung von Motoren

Die folgenden Terme wurden eingefügt, um Restriktionen an die Bewegung der Motoren zu modellieren. Das Modell ist so aufgebaut, daß ein Motor nur dann in Bewegung gesetzt werden kann, wenn der entsprechende “Bewegungs”-Term True ist. Da das TSG 5 Sitzmotoren kontrolliert, wurden entsprechend die 5 Terme **tHor_bewegen**, **tSchalung_bewegen**, **tVorne_bewegen**, **tHinten_bewegen** und **tWinkel_bewegen** eingeführt. Diese Terme werden dann True wenn eine Reihe von Bedingungen erfüllt ist. Zunächst darf ein Motor nur dann in Bewegung gesetzt werden wenn der ihm zugeordnete Taster (modelliert über die Variable mSitz_x vgl. Abschnitt 3.2.2) betätigt wird (dies ist das Event, das die Bewegung initiiert). Weiter wird abgefragt, ob die monitorierte Variable mT_offen den Wert True hat, denn nur dann ist eine manuelle Sitzeinstellung erlaubt. Desweiteren wird die Batteriespannung überprüft, ob sie über einem bestimmten Schwellwert liegen, um sicherzustellen, daß sich die Motoren gleichmäßig bewegen. Hat ein Sitz in einer der Richtungen in die er

bewegt werden kann (vgl. Abschnitt 3.2.2) eine maximale bzw. minimale Auslenkung erreicht, dürfen die Motoren nicht weiter angesteuert werden bzw. die Bewegung nicht gestartet werden. Somit nehmen die Terme den Wert True nur dann an, wenn die Extrempositionen noch nicht erreicht sind.

Die Bewegung eines Motores wird dann abgebrochen, wenn der ihm zugeordnete Term vom Wert True nach False wechselt. Dies ist dann der Fall, wenn eine der oben genannten Bedingungen verletzt wird bzw. dann wenn eine Blockierung des Sitzes auftritt. Wie das Auftreten einer Blockierung im Modell ermittelt wird ist in Abschnitt 4.3 näher beschrieben.

3.6 Beispielhafter Auszug aus dem Modell

Im folgenden werden anhand von Eventtabellen der kontrollierten Variablen cSmot_h und des Terms tHinten_bewegen beispielhaft Zusammenhänge von monitorierten Variablen, Termen und kontrollierten Variablen erläutert. Die Zusammenhänge zwischen Tastern, Motoren und den Termen der manuellen Sitzeinstellung wurden durch die Grafik in Abschnitt 3.1 schon aufgezeigt. In der Eventtabelle zu cSmot_h sieht man in dem Event "@T(tHinten_bewegen=TRUE) AND m_Sitz_h=heben" die direkte Abhängigkeit der kontrollierten Variable cSmot_h von der monitorierten Variable m_Sitz_h sowie von dem Term tHinten_bewegen.

Name		Mode Class	
cSmot_h		Sitzeinstellung	
Modes	**Events**		
Tuer_offen	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = heben)')	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = senken)')	Never
Sitz_bewegen_1	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = heben)')	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = senken)')	@T(tHinten_bewegen = FALSE)
Sitz_bewegen_2	Never	Never	@T(tHinten_bewegen = FALSE)
cSmot_h' =	heben	senken	ruhen

In der Eventtabelle des Terms tHinten_bewegen sieht man im Event, das tHinten_bewegen zu TRUE macht, die direkte Abhängigkeit des Terms von mSitz_h (sowie von weiteren Fahrzeugparametern wie mBatt). Somit ist auch eine indirekte Abhängigkeit der kontrollierten Variable cSmot_h von m_Sitz_h über den Term tHinten_bewegen gegeben.

Name		
tHinten_bewegen		
	Events	
	(@T(mSitz_h = heben) WHEN (cSpos_h != Maximalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE)) OR (@T(mSitz_h = senken) WHEN (cSpos_h != Minimalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE))	@T(mSitz_h = ruhen) WHEN (cSmot_h!=ruhen) OR @T(cSpos_h = Maximalwert) OR @T(cSpos_h = Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_h!=ruhen)) AND (NOT @C(cSpos_h) WHEN (cSmot_h!=ruhen))) OR @T(mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT)
tHinten_bewegen' =	TRUE	FALSE

4 Besonderheiten des Modells

In diesem Kapitel wird die Realisierung von besonderen Anforderungen an das TSG erläutert. Dabei werden die dazu eingeführten Terme beschrieben und Zusammenhänge zwischen Variablen und Termen erläutert, die zur Realisierung der Anforderungen wichtig sind.

4.1 Realisierung von Constraints und Prioritäten

Dieser Abschnitt beschreibt spezielle Anforderungen aus der textuellen Spezifikation des TSGs, die die Ansteuerung der Motoren betreffen bzw. Anforderungen, die Prioritäten bei der Benutzung der Taster beschreiben. Weiter werden Prioritäten von Motoren beschrieben (bei der Ansteuerung über die Managementtaster) und Ausnahmefälle betrachtet (keine Speicherwerte vorhanden).

4.1.1 Batteriespannung / Tür offen / Fahrzeuggeschwindigkeit

Die Realisierung der Constraints, daß die Motoren nur dann in Bewegung gesetzt werden dürfen, wenn die Batteriespannung über einem bestimmten Schwellwert liegt bzw. bei der Ansteuerung über die Sitztaster die Tür geöffnet sein muß, wurde bereits in Abschnitt 3.5.2 erläutert. Um zu verhindern, daß das Modell seinen Zustand wechselt, wenn der Benutzer versucht die Motoren anzusteuern obwohl einer der Constraints verletzt ist, wird auch in den Mode Transition Tables abgefragt, ob die Batterie die nötige Betriebsspannung hat bzw. die Tür offen ist.

Das Überprüfen der Fahrzeuggeschwindigkeit wird dann wichtig, wenn der Benutzer die Sitzeinstellung über den Funksender (die Geschwindigkeit muß 0 sein) oder über das Benutzermanagement (Geschwindigkeit <= 5 km/h) vornimmt. Auch hier wird ein Moduswechsel in den Mode Transition Tables verhindert (das Modell bleibt also in seinem aktuellen Zustand). Weiterhin wird die Fahrzeuggeschwindigkeit in den Event Tables aller Motoren überprüft (d.h. in den Event Tables der kontrollierten Variablen cSmot_x vgl. Abschnitt 3.3), um zu verhindern, daß sich die Motoren in Bewegung setzen.

4.1.2 Maximal 2 laufende Motoren

Benutzt der Fahrer des Fahrzeuges die Sitztaster des TSG zur manuellen Einstellung des Sitzes, so ist eine wichtige Bedingung, daß sich immer nur maximal 2

Motoren gleichzeitig bewegen dürfen. Dieses Constraint wurde im Modell durch die Einführung von zwei Modi, nämlich Sitz_bewegen_1 und Sitz_bewegen_2, realisiert. Wird ein Taster des TSG gedrückt so wechselt der Modus von Tuer_offen nach Sitz_bewegen_1 und der entsprechende Motor wird angesteuert. Betätigt der Benutzer einen weiteren Taster wechselt das Modell in den Modus Sitz_bewegen_2. Wird nun noch ein dritter Taster betätigt, so wird die Ansteuerung des zugehörigen Motors nicht vorgenommen, da bereits zwei Motoren in Bewegung sind (vgl. Kapitel 2.3.1).

4.1.3 Prioritäten von Motoren bei Ansteuerung über das Benutzermanagement

Betätigt der Benutzer des TSGs einen der beiden Benutzermanagementtaster (mMgmt_1 oder mMgmt_2 vgl Kapitel 2.1.4) so erfolgt die Ansteuerung der zu bewegenden Motoren in einer bestimmten Reihenfolge (die Bewegungen der Motoren erhalten eine Priorität). So sind zunächst alle entspannenden Bewegungen durchzuführen, d.h. alle Positionsänderungen die zu einer Entspannung der Sitzposition führen und dann alle einschränkenden Bewegungen. Weiterhin gilt folgende Priorität der Motoren: cSmot_hor (Entfernung Lenkrad) > cSmot_w (Lehnwinkel) > cSmot_s (Schalung) > cSmot_v (Sitzfläche vorne) > cSmot_h (Sitzfläche hinten), d.h. die Bewegung des Motors der den Abstand zum Lenkrad bestimmt hat die höchste Priorität, der Motor der den hinteren Teil der Sitzfläche einstellt die niedrigste. Auch bei der Einstellung über das Benutzermanagement dürfen sich maximal 2 Motoren gleichzeitig bewegen (vgl. Kapitel 4.1.2 Einstellung über die Sitztaster).

Diese Restriktionen wurden im Modell ebenfalls über zwei Modi (Bm_Sitz_bewegen_entspannen, Bm_Sitz_bewegen_einschraenken) und mehrere Terme realisiert. Das Modell führt zunächst, falls nötig, alle entspannenden Bewegungen durch (im Modus Bm_Sitz_bewegen_entspannen) und erst wenn alle entspannenden Bewegungen abgeschlossen sind und noch einschränkende Bewegungen durchzuführen sind wechselt das Modell in den Modus Bm_Sitz_bewegen_einschraenken. Um zu überprüfen, ob eine oder mehrere entspannenden bzw. einschränkenden Bewegungen durchzuführen sind wurden die Terme tBM_einschraenken_hor, tBM_einschraenken_w, tBM_einschraenken_s, tBM_einschraenken_v, tBM_einschraenken_h bzw. tBM_entspannen_hor, tBM_entspannen_w, tBM_entspannen_s, tBM_entspannen_v und tBM_entspannen_h eingeführt.

Sind entspannende Bewegungen durchzuführen d.h. das Modell wechselt in den Modus Bm_Sitz_bewegen_entspannen so stellen die entsprechenden Terme auch die Einhaltung der oben beschriebenen Prioritäten sicher. Zum Beispiel kann der Term tBM_einschraenken_s erst True werden, wenn die beiden Bewegungen mit höherer Priorität abgeschlossen sind, d.h. der entsprechende Term tBm_entspannen_hor bzw. w False ist. Somit wird auch sichergestellt, daß sich maximal zwei Motoren bewegen. Wird einer der Terme tBm_entspannen_x

(wobei x = h,v,s,w,hor) True so wird auch der zugehörige Motor (cSmot_x) in Bewegung gesetzt und zwar solange, wie der entsprechende Term auf True bleibt. Wechselt der Term nach False d.h. die einzunehmende Position ist erreicht, wird der Motor gestoppt. Erst wenn alle Terme tBm_entspannen_x False sind kann das Modell in den Modus Bm_Sitz_bewegen_einschraenken wechseln. Dort erfolgt die Einstellung des Sitzes analog zum entspannen (die Terme sind analog aufgebaut).

Durch diese Modellierung kann nun der Fall eintreten, daß sich nur ein Motor bewegt obwohl sich zwei Motoren bewegen könnten (ein Motor in entspannende und ein Motor in einschränkende Richtung). Dieser Fall kann deshalb auftreten, weil das Modell so aufgebaut ist, daß zuerst sämtliche entspannenden und erst dann sämtliche einschränkenden Bewegungen durchgeführt werden. Diese Modellierung führt zu einer weitaus geringeren Komplexität der SCR-Spezifikation, allerdings wird mehr Zeit zur Einstellung der Speicherpositionen benötigt (vgl. hierzu auch Anhang B, Frage 20)

4.1.4 Prioritäten der Taster

Die verschiedenen Taster des TSGs (Funksender, Benutzermanagement und die Sitztaster) unterliegen ebenfalls einer bestimmten Priorität. In der Spezifikation wurde nur die Restriktion gefordert, daß bei Funksender und Benutzermanagement der zuletzt gedrückte Taster die höhere Priorität besitzt. Im Modell wurde diese Anforderung derart erweitert, daß stets der zuletzt gedrückte Taster (unabhängig davon welcher Taster dies ist d.h. Sitztaster, Funksender oder Benutzermanagement) die höchste Priorität hat.

Diese erweiterte Anforderung wurde im Modell durch verschiedene Übergänge in die entsprechenden Modi realisiert bzw. durch entsprechende Einträge bei den Motoren (cSmot_x). Befindet sich das Modell im Modus Sitz_bewegen_1 oder 2 und der Benutzer drückt den Funksender bzw. einen Taster des Benutzermanagements, so wechselt das Modell in den entsprechenden Modus (BM_Funksender bzw. Sitz_bewegen_entspannen oder Sitz_bewegen_einschraenken, vgl. Abschnitt 3.4.1und Abschnitt 4.1.3). Wechselt das Modell den Modus muß dafür gesorgt werden, daß die zuvor initiierten Motorbewegungen gegebenen falls abgebrochen werden (z.B. wenn sich der Motor im neuen Zustand aufgrund von Prioritäten nicht bewegen darf). Im Modell wurden hierzu in den Event-Tables der Motoren entsprechende Einträge vorgenommen (vgl. Anhang C Tabelle 15 -18).

4.1.5 Keine Werte im Speicher

Der Speicher des TSGs, der die Sitzpositionen bestimmter Benutzer hält, ist bei der Inbetriebnahme des TSG noch nicht initialisiert. Betätigt ein Benutzer die

Benutzermanagement-Taster (mMgmt_1 bzw. 2) oder den Funksender (mFunksender_Pos1 bzw. 2) bevor Werte im Speicher abgelegt wurden, soll das TSG diese Events ignorieren, d.h. es erfolgt keine Reaktion. Nur wenn der Benutzer bereits Werte im Speicher abgelegt hat ist es möglich diese über die oben genannten Taster abzurufen. Um zu ermitteln, ob schon Sitzpositionen im Speicher abgelegt sind wurden die Variablen cWerte_gespeichert_1 und cWerte_gespeichert_2 (vgl. Abschnitt 3.3.6) eingeführt. Erst wenn diese Variablen den Wert True besitzen ist es möglich Speicherpositionen abzurufen. Da das Modell so realisiert wurde, daß zwei Benutzereinstellungen gespeichert werden können, benötigt man für jeden Speicher eine eigene Variable. Die Variablen werden auf True gesetzt, sobald der Benutzer die Taste zum Speichern von Werten in Kombination mit einer der mMgmt_x Tasten betätigt (mMgmt_set). Welche der beiden Variablen auf True gesetzt wird, bestimmt die Managementtaste die somit angibt, an welcher Speicherposition die Einstellungen abgelegt werden sollen.

4.2 Modellierung der Schnittstellen zur Umgebung

Im folgenden Kapitel werden die Sensoren und Aktuatoren beschrieben, die direkt am Sitz angebracht sind bzw. mit denen das TSG direkt interagiert. Von weiteren Sensoren, wie etwa die Sensoren zur Messung der Fahrzeuggeschwindigkeit oder der Batteriespannung, wird hier abstrahiert. Die Beschreibung der Sensoren und Aktuatoren wurde in der Spezifikation des TSGs auf einer sehr technischen Ebene vorgenommen von der im Modell ebenfalls abstrahiert wurde. Weiter werden in diesem Kapitel die Wertebereiche der Variablen die Schnittstellen modellieren erläutert.

4.2.1 Modellierung der Sitzsensoren

Zu den Sitzsensoren zählen die Sensoren, die die aktuelle Position des Sitzes bereitstellen und solche die die Taster repräsentieren, d.h. die Bewegungsanforderungen registrieren.

Die Sitztaster werden in der Spezifikation über ein Widerstandsnetzwerk beschrieben. Im Modell wurde von dieser sehr technischen Beschreibung abstrahiert. Die Stellung der Taster wird nicht über Widerstandwerte beschrieben sondern mit natürlichsprachlichen Bezeichnungen für die jeweilige Tasterstellung. Dadurch sind die Wertebereiche der monitorierten Variablen mSitz_x (vgl. Abschnitt 3.2.2) leichter verständlich (z.B heben / senken / ruhen).

Auch die aktuelle Position des Sitzes wird in der Spezifikation über Widerstandswerte beschrieben (zwischen 1k und 10k Ohm). Dieser Wertebereich wurde ebenfalls übertragen auf einen Wertebereich der von der Ohmzahl abstrahiert. So haben die kontrollierten Variablen cSpos_x (die die Sensoren zur Positionsan-

gabe modellieren) einen Wertebereich von 0 bis 10. Dabei repräsentiert die 0 die minimale und 10 die maximal Auslenkung des Sitzes in der entsprechenden Richtung.

Die Taster des Benutzermanagements (mMgmt_x vgl. Abschnitt 3.2.3) können im Modell lediglich die Werte True oder False annehmen. Dies entspricht der technischen Beschreibung die vorsieht, daß der Taster entweder gedrückt oder nicht gedrückt ist.

4.2.2 Modellierung der Sitzaktuatoren

Zu den Sitzaktuatoren zählen die Sitzmotoren, die vom TSG angesteuert werden. Diese wurden über die kontrollierten Variablen cSmot_x (vgl. Abschnitt 3.3.3) modelliert. Auch hier wurde von der technischen Beschreibung abstrahiert, die die Bewegungsrichtung der Motoren über Voltzahlen beschreibt. Das Modell verwendet zur Richtungsangabe natürlichsprachliche Bezeichnungen, die denen für die Taster die den Motor kontrollieren entsprechen (d.h. der Motor kann z.B. die Werte heben senken und ruhen annehmen).

Die Geschwindigkeit in der ein Motor die Position des Sitzes verändern soll wurde auf den Wertebereich der Sitzpositionen normiert. Pro Zeiteinheit verändert sich die Position des Sitzes um eine Einheit.

4.3 Die Umgebung des Türsteuergerätes: Sitz-Blockierung

In diesem Abschnitt werden Ereignisse beschrieben, die in der Umgebung des TSGs auftreten können, aber nur indirekt beobachtbar sind. Im Modell wurde als einziges Event dieser Art, die Blockierung des Fahrersitzes modelliert. Die Beschreibung der Realisierung einer Blockierung ist Gegenstand dieses Abschnitts.

Bei der Modellierung einer Blockierung des Sitzes ist folgendes zu beachten. Da es nicht möglich ist mit Hilfe von SCR ein Ereignis zu modellieren, das eine Blokkierung des Sitzes auslöst, wurden im Modell pro Motor je eine Variable eingeführt, die ein solches Ereignis modelliert (mBlockiert_x vgl. Abschnitt 3.2.1).

Das eigentliche Überprüfen, ob eine Blockierung eines Motors vorliegt wird allerdings innerhalb des Modells vorgenommen. Die Spezifikation sieht vor, daß ein Sitzmotor dann als blockiert anzusehen ist, wenn sich innerhalb einer bestimmten Zeiteinheit die entsprechende Position des Sitzes nicht verändert hat. Ist dies der Fall, so ist der blockierte Motor in den Zustand Ruhen zu versetzen. Die oben genannten Variablen werden dazu verwendet, eine Änderung der Position zu verhindern, d.h. ist eine der Variablen auf True gesetzt, so verändert sich die entsprechende Position nicht, obwohl der Motor läuft. In den Event-

Tables der Motoren wird mit Hilfe eines Change Events überprüft, ob sich die dem Motor zugeordnete Position verändert. Ist dies nicht der Fall und der Motor befindet sich nicht im Zustand Ruhen, liegt eine Blockierung vor und die Bewegung des Motors wird abgebrochen.

Speziell für die automatische Einstellung des Sitzes über den Funksender oder das Benutzermanagement wurden Terme eingefügt, die ebenfalls mit Hilfe eines Change-Events prüfen ob sich die Position verändert oder nicht. Diese Terme (tBM_Blockierung_aufgetreten_h, tBM_Blockierung_aufgetreten_v, tBM_Blockierung_aufgetreten_s, _Blockierung_aufgetreten_hor, tBM_Blockierung_aufgetreten_w) werden genau dann True, wenn sich der Motor in Bewegung befindet und die Position nicht geändert wird. Ist einer der Terme True wird der entsprechende Motor abgeschaltet, andere nichtblockierte Motoren können sich noch weiter bewegen.

5 Erfahrungen und wünschenswerte SCR Erweiterungen

In diesem Kapitel werden zuerst Erfahrungen, die mit SCR während der Modellierung gemacht wurden dokumentiert. Danach werden wünschenswerte Erweiterungen von SCR genannt, die die Modellierung erleichtert hätten.

5.1 Erfahrungen

1. Wir haben bei der Modellierung anfangs nicht beachtet, daß ein Taster z.B. MGMT_1 unter Umständen mehrere Funktionen wahrnimmt (In diesem Fall Speicherposition abrufen bzw. dann abspeichern). Daher hatten wir den Taster nur eingesetzt um die Motoren anzusteuern was zu einem Fehlverhalten führt wenn abgespeichert werden soll. D.h. durch das Einführen der neuen Funktion mußten wir die alte überarbeiten.

   Ratschlag: Man sollte sich nach dem Identifizieren der monitorierten Variablen und deren Abhängigkeiten überlegen welche Funktionen jeweils von einer bestimmten monitorierten Variable abhängen.Dadurch vermeidet man, daß man nachdem man eine Funktion implementiert hat Bedingungen vergißt und diese dann nachträglich einfügen muß.

2. Wir haben die Modellierung sehr stark an Zustandsdiagramme angelehnt. So haben wir bei vielen Zustandsübergängen auch Aktionen ausgeführt und nicht erst einen Zustand angenommen und dann die entsprechende Aktion ausgeführt. Dadurch sind Eventtables der kontrollierten Variablen komplizierter ausgefallen. Und wir hatten eine Mischung aus Event- und Conditiontables im Modell.

   Ratschlag: Wenn möglich sollte man Conditiontables benutzen, d.h. Aktionen ausführen, wenn man in dem Mode drinnen ist, nicht beim Übergang der Modi.

3. Einen Teil der Funktionalität des Speicherns der aktuellen Sitzposition haben wir implizit modelliert. Normalerweise hätte man abfragen müssen ob MGMT_Set gedrückt ist, damit beim Drücken von MGMT_1 nicht die Motoren angeschaltet werden. Dies geschah dadurch, daß wir zunächst den MGMT_Set Taster drücken und dann MGMT_1 bzw. 2. Dadurch werden sofort die Speicherwerte auf die aktuelle Position gesetzt und somit fangen die Motoren nicht an zu laufen, da der Abgleich von Sitzposition und Speicherposition immer TRUE ergibt.

4. Wir führten Terme "ad hoc" ein, zur besseren Lesbarkeit und wenn wir gemerkt haben, daß man den Term mehr als einmal braucht. Eine systematischere Vorgehensweise bei der Termbildung wäre sicher sinnvoller gewesen.

   Ratschlag: Man sollte sich frühzeitig Gedanken machen, welche Terme man im Laufe der Modellierung häufig benötigt. Wenn man einen Term einführt, sollte man sich gleich überlegen an welchen Stellen man ihn später noch verwenden kann und ihn ggf. so anpassen, daß man ihn an mehreren Stellen benutzen kann.

5. Das Modell wurde inkrementell entwickelt. Von Schritt zu Schritt wurden neue Anforderungen hinzugenommen. Eine andere Partitionierung der Anforderungen wäre evtl. von Vorteil gewesen. Dadurch daß das Ansteuern der Motoren über das Benutzermanagement in einem späteren Schritt eingeführt wurde, aber relativ viel Gemeinsamkeit mit dem manuellen Ansteuern des Sitzes hat, sind Redundanzen in unserem Modell aufgetreten, die teilweise bemerkt und eliminiert wurden. Trotzdem gibt es weiterhin ähnliche Konstrukte in den Teilen der manuellen und automatischen (mittels Benutzermanagement oder Funksender) Sitzeinstellung.

5.2 Wünschenswerte SCR-Erweiterungen

1. Würde SCR eine Parametrisierung der Variablen unterstützen, hätten wir viele Variablen nicht in fünffacher Ausführung gebraucht (z.B. cSmot_h, cSmot_w, usw.). Dies hätte nicht nur Arbeit erspart, sondern auch die Lesbarkeit der Spezifikation erhöht.

2. Könnte man beim Simulator eine Umgebung anbinden (z.B. über Stubs), so hätten wir Variablen, die eigentlich zur Umgebung gehören, nicht in SCR modellieren müssen (z.B. cSpos_h, usw.). Prinzipiell wäre das zwar kein Problem, dadurch daß SCR aber keine Partitionierbarkeit zusammengehöriger Modellteile besitzt, mußten wir alles in einem Modell (in einer Modeklasse) modellieren. Dies machte die Modeklasse komplexer.

3. An den Stellen, wo wir mehrere logisch zusammenhängende Modi hatten, hat uns die Möglichkeit einer Hierarchiebildung in den Modeklassen ähnlich einer Zustandsverfeinerung bei Zustandsdiagrammen gefehlt (z.B. die Modi Sitz_bewegen_1 und Sitz_bewegen_2 hätte man als Verfeinerung des Modus' Sitz_bewegen modellieren können). Diese hätte die Modellierung vereinfachen können.

4. Eine Suchen und Ersetzen Funktion ist eigentlich in jedem Programm unersetzlich. Gerade wenn Variablen oder Konstanten umbenannt werden mußten, war es sehr umständlich und fehlerträchtig dies von Hand (mit Hilfe eines Texteditors) zu tun. Weiterhin hätten wir mit einer Suchfunktion schnell alle Stellen im Modell gefunden, an der eine Variable genutzt wird.

6 Entwurfsentscheidungen und Besonderheiten des Modells

In diesem Kapitel werden Entscheidungen, die während des Modellierung getroffen wurden erläutert. Weiterhin werden Besonderheiten des Modells aufgezeigt.

6.1 Entwurfsentscheidungen

Laut Spezifikation dürfen bei der Sitzeinstellung mittels Benutzermanagement max. 2 Motoren gleichzeitig laufen. In der Modeklasse Sitzeinstellung unterscheiden wir deshalb zwischen entspannen- und einschränken- Modi, d.h. zunächst werden alle entspannenden Sitzbewegungen ausgeführt, dann die einschränkenden. Dadurch kann es vorkommen, daß nur 1 Motor läuft (zur Entspannung) obwohl noch ein anderer laufen könnte (aber zum einschränken). Dadurch wurde die Modellierung weniger komplex.

Wir haben nur 2 Speicherpositionen modelliert, nicht 4, da sich keine neuen Erkenntnisse aus der Modellierung weiterer Speicherpositionen ergeben hätten und eine Modellierung in SCR aufwendiger gewesen wäre.

Variablen, die im internen Speicher des TSGs liegen (z.B. die Sitzpositionen) haben wir als kontrollierte Variablen bzw. als Konstanten (sofern es sich um Konfigurationsparameter handelt, die sich nicht ändern) modelliert.

In der Spezifikation des TSG wird auf Seite 33 beschrieben, daß bei einem Timeout der Can-Botschaft mIn_Fzg_v ein Fehlereintrag gesetzt werden muß und angenommen wird, daß die Geschwindigkeit 10km/h beträgt. Da dieser Aspekt durch den Canbus kommt und kein Charakteristikum des TSG ist, wurde dies nicht modelliert. Es wird davon ausgegangen, daß die Fahrzeuggeschwindigkeit korrekt übermittelt wird.

6.2 Besonderheiten

Dieser Abschnitt beschreibt Besonderheiten bei der Benutzung des Simulators (aus dem SCR Toolset) und ein Fehler, der sich noch im Modell befindet.

6.2.1 Benutzung der Eventscripts im SCR-Simulator

Bevor man die Funktionen des Benutzermanagements bzw. des Funksenders testet, müssen im TSG erst mal Werte abgespeichert sein. Da die Simulation des TSG aber aus dem Initialzustand heraus durchgeführt wird, sind eigentlich noch keine Werte vorhanden, was in der Modellierung durch den Wahrheitswert FALSE der Variablen cWerte_gespeichert_1 bzw. cWerte_gespeichert_2 ausgedrückt wird. Dies bedeutet, daß in allen Eventscripts, die auf vorher abgespeicherten Werten beruhen erst einmal Speicherpositionen besetzt werden muessten. Dies erfordert viele Zeiteinheiten vorweg, bevor die eigentliche Funktion ausgeführt werden kann. Daher schlagen die Autoren dieses Berichtes folgende Vorgehensweise vor:

1. Im SCR-Simulator den Editmode einschalten

2. cWerte_gespeichert_1 und cWerte_gespeichert_2 auf TRUE setzen

3. Im SCR-Simulator den Editmode verlassen

4. Eventscript zu Benutzermanagement bzw. zu Funksender laden und ausführen

Eventscripts, die keine Speicherwerte benötigen (z.B. manuelle Sitzeinstellung) sind von diesem Problem nicht betroffen. Hier können die Eventscripts direkt geladen werden.

6.2.2 Abweichungen des Modells von der textuellen Spezifikation

In dem Modell steckt noch eine Abweichung von der textuellen Spezifikation, die den Autoren bekannt ist, aber nicht behoben ist. Im folgenden wird das Problem kurz beschrieben und ein Lösungsvorschlag vorgestellt.

Problembeschreibung: Bei der Modellierung mit SCR wurde davon ausgegangen, daß Events nur dann gleichzeitig (parallel) auftreten können, wenn sie durch die gleiche Variable ausgelöst werden. D.h. das Ändern einer Variable kann zu zwei Events führen, das Ändern von 2 Variablen und die dadurch ausgelösten Events werden sequentialisiert. Diese Vorstellung gilt aber nur für monitorierte Variablen. Ändern sich zwei kontrollierte Variablen gleichzeitig, so können auch Events, die dadurch ausgelöst werden parallel auftreten. In der Modellierung des TSG ist dies dann der Fall, wenn sich zwei Sitze bewegen und die Extremwerte gleichzeitig erreicht werden (z.B. cSpos_h und cSpos_v sind jeweils 5 und werden für 5 Zeiteinheiten inkrementiert. Dadurch erreichen beide gleichzeitig die Maximalauslenkung 10). Normalerweise, wenn die Events sequentialisiert würden, würde das TSG vom Mode Sitz_bewegen_2 erst in den Mode Sitz_bewegen_1 wechseln um dann in Ruhen bzw. Tuer_offen zu wech-

seln. So lösen beide Events gleichzeitig aus und das TSG geht von Sitz_bewegen_2 in Sitz_bewegen_1 und bleibt dort. Dies führt dann bei der folgenden Bedienung zu Fehlfunktionen.

Lösungsvorschlag: Man kann einen Zustandsübergang direkt von Sitz_beweegen_2 nach Ruhen bzw. Tuer_offen einführen. Alle Möglichkeiten des gleichzeitigen Erreichens der Maximalauslenkungen könnten dann mit einem Boolschen Ausdruck beschrieben werden, der dann sowohl als Event für den Zustandsübergang von Sitz_bewegen_2 nach Ruhen bzw. Tuer_offen benutzt wird, als auch als Bedingung um den Übergang von Sitz_bewegen_2 nach Sitz_bewegen_1 zu verhindern (um Determinismus zu gewährleisten).

7 Beschreibung der Event-Scripts

Dieses Kapitel beschreibt die während der Modellierung erstellten Eventscripts. Diese wurden hauptsächlich zur Simulation des idealen Systemverhaltens eingesetzt bzw. wurden zur Prüfung von Ausnahmesituationen eingesetzt.

Folgende Event-Scripts wurden erstellt:

- Manuelle Einstellung úber die Taster:
  3_Taster_gedrueckt
  BatterieSchwellwert_unterschritten
  Blockierung_hMotor
  Normalfall_und_Grenze
  Tuer_ungeoeffnet
  Tuerschliessen_waehrend_Bedienung

- Funksender:
  Einstellung_ueber_Funksender
  Zwei_Funksendertaster_betaetigt
  Funksender_Normaler_Tatser
  Funksender_Blockierung
  Funksender_Batt_zu_niedrig
  Funksender_FZG_V_erhoet
  Funksender_FZG_V_zu_gross
  Funksender_BM_Tastrer_gedrueckt

- Benutzermanagement
  Alle_Motoren_entspannen_nach_Prioritaet
  Erst_entspannen_dann_einschraenken
  Letzter_BM_Taster_gedrueckt_hoehere_Prio
  Normaler_Taster_hoehere_Prio
  BM_Taster_hoehere_Prior
  FZG_V_zu_gross_keine_Bew
  FZG_V_wird_zu_gross_Bew_abbruch
  Speichern_aktuelle_Position
  Blockierung

Die folgenden Abschnitte (Abschnitt 7.1, Abschnitt 7.2, Abschnitt 7.3) erläutern die einzelnen Event-Scripts. Vor der Benutzung der Event-Scrips für das Benutzermanagement sei auf Abschnitt 6.2.1 verwiesen. Dort werden wichtige Hin-

weise gegeben, die bei der Benutzung des Simulators mit den Events-Scripts fúr das Benutermanagement zu beachten sind

7.1 Event-Scripts zur manuellen Einstellung über die Taster

1. Normalfall_und_Grenze
   Events:
   Es werden zwei Taster zur manuellen Sitzansteuerung betätigt. Die Taster werden losgelassen und andere Taster betätigt.

   Reaktionen:
   Beim ersten Betätigen der Taster werden die entsprechenden Motoren angesteuert und beim loslassen der Taster stoppen die Motoren. Beim zweiten Betätigen der Taster beginnen die entsprechenden Motoren zu laufen. Einer dieser Motoren erreicht seine minimale Auslenkung und daher wird die Bewegung abgebrochen.

2. BatterieSchwellwert_unterschritten
   Events:
   Es wird ein Taster gedrückt. Während der Ansteuerung wird der Batterieschwellwert unterschritten. Danach wird der Taster losgelassen und ein zweites mal gedrückt. Der Taster wird wieder losgelassen, der Batteriesachwellwert wieder überschritten und der Taster zum dritten mal gedrückt.

   Reaktionen:
   Nach dem ersten Tastendruck, beginnt der entsprechende Motor zu laufen. Sobald der Batterieschwellwert unterschritten wird, bricht die Motorbewegung ab. Beim zweiten Drúcken des Tasters wird der Motor nicht angesteuert. Beim dritten Drücken bewegt er sich wieder.

3. Blockierung_hMotor
   Events:
   Es werden zwei Taster zur manuellen Sitzansteuerung betätigt. Während der Ansteuerung wird einer der Motoren blockiert.

   Reaktionen:
   Der blockierte Motor wechselt in den Zustand “Ruhen”, d.h. die entsprechende Bewegung wird abgebrochen.

4. 3_Taster_gedrueckt
   Events:
   Es werden nacheinander drei Taster zur manuellen Ansteuerung gedrückt und nacheinander wieder losgelassen.

   Reaktionen:

Es werden lediglich die zwei Motoren angesteuert, deren Taster zuerst gedrúckt wurden. Der dritte Taster wird ignoriert.

5. Tuer_ungeoeffnet
   Events:
   Die Tür ist nicht geöffnet und ein Taster zur manuellen Sitzansteuerung wird betätigt.

   Reaktionen:
   Der Tastendruck wird ignoriert, der entsprechende Motor wird nicht angesteuert.

6. Tuerschliessen_waehrend_Bedienung
   Events:
   Ansteuerung des Sitzes mittels eines Tasters zur manuellen Sitzeinstellung. Während dessen wird die Tür geschlossen.

   Reaktionen:
   Wird die Tür während der Ansteuerung geschlossen, wird die Motorbewegung abgebrochen.

7.2 Event-Scripts zur Ansteuerung mittles Funksender

1. Einstellung_über_Funksender
   Events:
   Einstellung des Sitzes durch den Funksender

   Reaktionen:
   Alle Motoren bewegen sich, bis die entsprechende Speicherposition erreicht ist.

2. Zwei_Funksender_Taster_betaetigt
   Events:
   Zunächst Einstellung mittels des ersten Funksendertaster.
   Während der Einstellung wird der zweite Funksendertaster betätigt.

   Reaktionen:
   Zunächst bewegen sich alle Motoren, um die Sitzposition der ersten Speicherposition einzunehmen. Wird der zweite Taster betätigt hat dieser höhere Priorität und die Motoren bewegen den Sitz in die zweite Speicherposition

3. Funksender_Normaler_Taster
   Events:
   Ansteuerung über den Funksendertaster
   Benutzer öffnet die Fahrzeugtúr und drückt einen manuellen Taster

Reaktionen:
Der Taster zur manuellen Ansteuerung hat die höhere Priorität. Alle Motorbewegungen die durch den Funksendertaster ausgelóst wurden werden abgebrochen, bis auf die Bewegung des Motors, dessen Taster betätigt wurde

4. Funksender_Blockierung
   Events:
   Ansteuerung über den Funksendertaster
   Während der Bewegung tritt eine Blockierung eines Motors auf

   Reaktionen:
   Der Blockierte Motor geht in den Zustand "Ruhen", alle anderen (nicht blokkierte) Motoren werden weiter angesteuert, bis entsprechende Speicherposition erreicht ist.

5. Funksender_Batt_zu_niedrig
   Events:
   Ansteurerung úber den Funksendertaster, wáhrend dessen wird der Batterieschwellwert unterschritten

   Reaktionen:
   Alle angestossenen Bewegungen werden abgebrochen, das Modell geht in den Zustand "Ruhen" bzw. "Tuer_offen"

6. Funksender_FZG_V_zu_gross
   Events:
   Die Fahrzeuggeschwindigkeit ist grösser 0 und der Funksendertaster wird betätigt.

   Reaktionen:
   Das Funksendersignal wird ignoriert, es werden keine Motoren angesteuert.

7. Funksender_FZG_V_erhoeht
   Events:
   Ansteuerung über den Funksendertaster. Während der Ansteuerung wird das Fahrzeug in Bewegung gesetzt (d.h. FZG_V > 0)

   Reaktionen:
   Alle Motorbewegungen werden abgebrochen, das Modell geht in den Zustand "Ruhen"

8. Funksender_BM-Taster_gedrueckt
   Events:
   Ansteuerung über den Funksendertaster. Dann wird ein Benutzermanagementtaster betätigt.

Reaktionen:
Der Benutzermanagementtaster hat höhere Priorität. Alle Motoren gehen in den Zustand "Ruhen", bis auf die Motoren, die durch den Benutzermanagementtaster angestossen wurden.

7.3 Event-Scripts zur Ansteuerung mittels Benutzermanagementtaster

1. Alle_Motoren_entspannen_nach_Priorität
   Events:
   Ansteuerung über einen Benutzermanagementtaster, wobei nur entspannende Bewegungen auszuführen sind

   Reaktionen:
   Die Motoren werden entsprechend ihrere Prioritäten (zuerst hor dann, w,s,v,h) angesteuert.

2. Erst_entspannen_dann_einschränken
   Events:
   Ansteuerung mittels eines Benutzermanagementtatsers, wobei nun sowohl entpannende als auch einschränkende Bewegungen durchzuführen sind

   Reaktionen:
   Zunächst werden die Motoren entsprechend ihrere Priorität angesteuert um die entsoannenden Bewegungen durchzuführen. Sind diese Bewegungen alle abgeschlossen, wechselt das Modell in den Zustand "BM_sitz_bewegen_einschraenken". Dort werden die Motoren entsprechend ihrer Prioriät angesteuert um die einschrämkende Bewegung durchzuführen.

3. Letzter_BM_Taster_gedrueckt_hoehere_Prio
   Events:
   Ansteuerung mittels des ersten Benutzermanagementtasters, dann drücken des zweiten Benutzermanagementtasters.

   Reaktionen:
   Der zuletzt gedrúckte Taster hat höhere Priorität. Alle Bewegungen die durch den ersten Taster initiiert wurden werden abgebrochen.

4. Normaler_Taster_hoehere_Prio
   Events:
   Ansteurerung mittels Benutzermanagementtaster. Nach einigen Schritten wird die Tür geöffnet und der Benutzer betátigt einen Taster zur manuellen Sitzeinstellung

Reaktionen:
Der Taster zur manuellen Einstellung hat höhere Priorität. Alle Motorbewegungen werden abgebrochen, mit Ausnahme des Motors, der dem betätigten Taster zugeordnet ist. Dieser nimmt den Zustand entsprechend der Tasterstellung ein (heben oder senken)

5. BM_Taster_hoehere_Prior
   Events:
   Ansteuerung über einen Taster zur manuellen Sitzeinstellung. Dann betätigen eines Benutzermanagementtasters.

   Reaktionen:
   Der Benutzermanagementtaster hat höhere Priorität. Das Modell verhält sich entsprechend den Anforderungen zur Einstellung des Sitzes mittels Benutzermanagementtaster.

6. FZG_V_zu_gross_keine_Bew
   Die Fahrzeuggeschwindigkeit ist grösser als der zulässige Schwellwert (5 km/h). Der Benutzer betätigt einen Benutzermanagementtaster

   Reaktionen:
   Es werden keine Motoren angesteuert, der Tastendruck wird also ignoriert.

7. FZG_V_wird_zu_gross_Bew_abbruch
   Events:
   Ansteuerung mittels Benutzermanagementtaster, während dessen steigt die Fahrzeuggeschwindigkeit über den zulässigen Schwellwert

   Reaktionen:
   Alle Bewegungen werden abgebrochen, das Modell wechsel in den Zustand "Ruhen"

8. Speichern_aktuelle_Position
   Events:
   Bewegung des Sitzes mittels der Taster zur manuellen Ansteuerung. Dann Abspeichdern der eingenommenen Position.

   Reaktionen:
   Die Speicherwerte nehmen die Werte der aktuellen Sitzposition an, die Variable cWerte_gepeichert_1 oder 2 wird auf true gesetzt (je nach betätigtem Benutzermanagementtaster)

9. Blockierung
   Events:
   Ansteuerung mittels Benutzermanagementtaster. Dabei tritt eine Blockierung eines Motos auf.

Reaktionen:
Die Bewegung des blockierten Motors wird abgebrochen. Alle anderen Motoren werden weiter angesteuert, bis die angeforderte Sitzposition erreicht ist.

8 Zusammenfassung

Im vorliegenden Bericht wurde die Modellierung des TSG mit der Modellierungssprache SCR beschrieben. Es liegt jetzt eine Ausgangsbasis vor, um die Eignung von SCR-Modellen als Teil des Systemspezifiaktionsdokuments zu bewerten.

Grundsätzlich erscheint SCR sehr geeignet, da die Sprache eine sehr einfache Semantik hat und die tabellarische Darstellung eine sehr kompakte Darstellung erlaubt. Im konkreten Fall des TSG hat die fehlende Parametrisierbarkeit dazu geführt, daß das Modell etwa fünfmal größer ist, als im Falle einer parametrisierbaren Sprache. Der Grund ist, daß jedes Systemverhalten fünfmal, d.h. für jeden Motor des Sitzes, modelliert werden mußte. Mit der Größe sinkt natürlich die Lesbarkeit und Verständlichkeit des Modells. Bei den bislang weggelassenen Außenspiegeln wird das gleiche Problem auftreten.

Das vorliegende TSG-Modell beschreibt das Fahrer-TSG. Das Beifahrer-TSG ist sehr ähnlich, d.h. durch geschickte Parametrisierung könnte ein generisches TSG spezifiziert werden, um z.B. die Kommunikation zwischen Fahrer- und Beifahrer-TSG zu simulieren. Momentan müßte dazu das komplette Modell dupliziert werden, was im Hinblick auf Änderungen nicht sinnvoll erscheint.

Zusammenfassend läßt sich sagen, daß SCR ungeeignet zur Modellierung von Systemen mit sehr ähnlichen oder gleichen Teilsystemen ist. Denkbar wäre aber eine Art "SCR-Preprozessor" der ein parametrisierte SCR-Spezifikation in eine nicht-parametrisierte umwandelt, ähnlich wie ein C++Preprozessor C++Kode nach C-Kode umwandelt.

Anhang A Introduction to SCR

The Software Cost Reduction (SCR) requirements specification technique was developed for the domain of reactive systems. SCR is today a formal description technique for system and software requirements of reactive systems. The technique was developed by Parnas et al. two decades ago at the Naval Research Laboratory as part of the Software Cost Reduction project to specify the operational flight program for the Navy's A-7 aircraft concisely and unambiguously [Hen80]. Heitmeyer et al. strengthened SCR's formal foundations by adding a formal semantics [HJL96] and has provided appropriate tool support known as *SCR toolset* [HBGL95].

A reactive system is represented in SCR as a finite state automaton. The states are called *modes* and transitions are triggered by events. The required system behavior is described by a few constructs: monitored and controlled variables, modes, mode classes, terms, conditions, and events. A *monitored variable* represents an environmental quantity that influences system behavior and a *controlled variable* represents an environmental quantity that the system controls. A *mode* is a subset of the system's state space. A *mode class* is a composition of modes such that the modes of a mode class are mutually exclusive and exhaustive; it is a state machine. A *term* is a notational short-hand for a complex expression on monitored variables, modes, or other terms. A *condition* is a predicate on monitored variables, modes, or terms. An *event* occurs when a monitored variable, term, or mode class changes its value. Mode classes are defined by *mode transition tables* and terms and controlled variables are defined by *event tables* or *condition tables*. A *dependency graph* shows the "uses value" dependencies among monitored variables, terms, mode classes, and controlled variables.

In order to illustrate the notational elements of SCR, we use the ESFAS requirements. The ESFAS (Engineered Safety Feature Actuation System) requirements were originally introduced by Courtois and Parnas [CP93]. The ESFAS, as used in Belgian and Canadian PWR type nuclear power plants, prevents or mitigates damage to the core and coolant system when a fault such as loss of coolant occurs. The ESFAS, depicted in Figure 1. Engineered Safety Feature Actuation System (ESFAS)Figure 1. Engineered Safety Feature Actuation System (ESFAS), receives data from some pressurizer's pressure sensors and determines whether an actuation signal called 'safety injection' must be sent to the safety feature components that cope with pressure accidents. The human ESFAS operator has two push buttons to block the safety injection signal and to reset the blockage.

We have simplified the ESFAS requirements slightly. Rather than the original three independent sensors and a voting mechanism, we assume a single sensor that measures the pressurizer's pressure. The ESFAS requirements are:

+ UR1The system monitors the pressure and sends the safety injection signal when the pressurizer's pressure falls below a 'low' threshold.
+ UR2The human operator can override system actions by turning on a 'Block' button and resets the manual block by pushing on a 'Reset' button.
+ UR3A manual block is permitted if and only if the pressure is below a 'permit' threshold.
+ UR4The manual block must be automatically reset by the system after 15 minutes.
+ UR5A manual block is effective if and only if it is executed before the safety injection signal is sent.
+ UR6The 'Reset' button has higher priority than the 'Block' button.

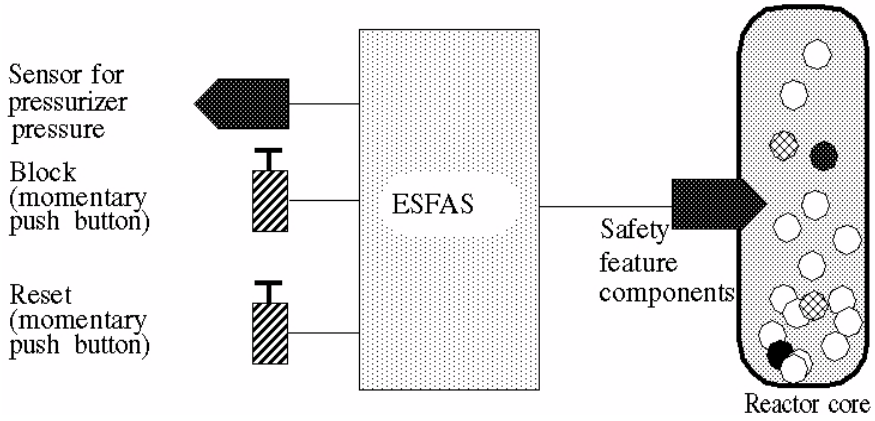


Figure 1. Engineered Safety Feature Actuation System (ESFAS)

The original set of requirements was incomplete, UR2, UR4, and UR6 supply information that was missing.

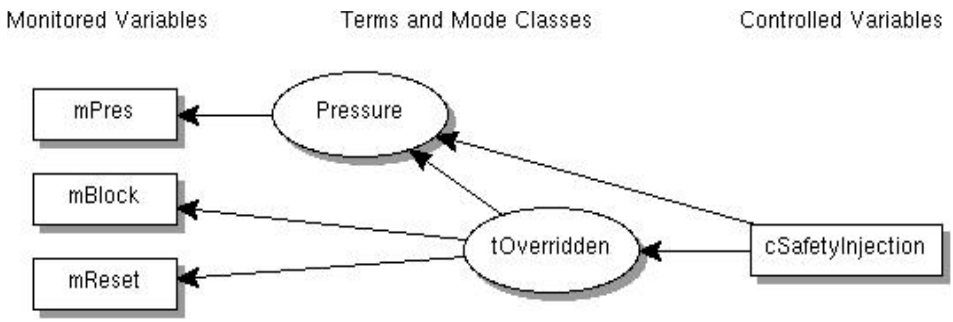


Figure 2. Dependency Graph for ESFAS Requirements

Figure 2. Dependency Graph for ESFAS Requirements shows the dependency graph for the ESFAS. The boxes are variables, and the blobs are mode classes or terms. The arrows indicate, which variable, mode class, or term depends in its value on which other.

Table 1: Mode Transition Table of ‘Pressure’ (UR1, UR3

Old Mode	Event	New Mode
TooLow	@T(mPres ≥ Low)	Permitted
Permitted	@T(mPres ≥ Permit)	High
Permitted	@T(mPres < Low)	TooLow
High	@T(mPres < Permit)	Permitted

Table 2.: Event Table for ‘tOverridden’ (UR2, UR3, UR4, UR6)

Mode	Events	
High	False	@T(Inmode)
TooLow, Permitted	@T(mBlock = On) WHEN (mReset = Off)	@T(Inmode)[a] OR @T(mReset = On) OR @T(Duration2(tOverridden = True) > 15)
tOverridden =	True	False

a. Note that this event occurs when (TooLow OR Permitted) becomes true, i.e. it occurs when the mode changes from ‘High’ to ‘Permitted’. This event does not occur when the mode changes from ‘Permitted’ to ‘TooLow’ or vice versa.

Table 3: Condition Table for ‘cSafetyInjection’ (UR1)

Mode	Conditions	
High, Permitted	True	False
TooLow	tOverridden	NOT tOverridden
cSafetyInjection =	Off	On

Table 3: Condition Table for ‘cSafetyInjection’ (UR1) show the definitions of the mode class ‘Pressure’ by a mode transition table, the term ‘tOverridden’ by an event table, and the controlled variable ‘cSafetyInjection’ by a condition table.

The initial mode of ‘Pressure’ is ‘Permitted’, the initial value of ‘tOverridden’ is ‘False’, and the initial value of ‘cSafetyInjection’ is ‘Off’. The constant ‘Low’ has the value 20 and the constant ‘Permit’ has the value 40. The requirements model is incomplete; requirement UR5 is not modeled for didactic reasons, because we want to illustrate all types of tables. UR5 would require ‘cSafetyInjection’ to be defined by an event table rather than a condition table.

The *environmental assertion* and the *specification assertion* are two SCR constructs that were introduced in conjunction with the SCR toolset. An environmental assertion defines a range of values in the domain of a monitored variable, which never naturally occur because of properties of the environment. A specification assertion defines invariants of the requirements model. Environmental and specification assertions are denoted by implications, e.g.,

mBlock=On => mReset=Off

which states that if the Block switch is pressed, the Reset switch cannot be pressed, perhaps, because of mechanical construction of the switches. Note this specification assertion conflicts with the ESFAS requirement UR6, because the former is an indicative statement, the latter is an optative statement, and both state that the Block button has higher priority. An example of a specification assertion is

@T(tOverridden) => cSafetyInjection = Off

which states that the safety injection can be overridden only if it is not already active. The given SCR requirements model of the ESFAS would violate this assertion, because UR5 is not modeled.

A Process for Developing SCR Requirements Models.

This process represents an agenda for creating SCR requirements models. We used as a basis for this agenda the process for developing a system requirements specification developed by a working group of a recent Dagstuhl seminar [BHPR99], and we tailored this process towards SCR. The agenda assumes that the SCR toolset is used. Rather than providing complete validation conditions, we provide only the name of the check as implemented in the consistency chekker of the toolset.

Table 4. Agenda for Creating SCR Requirements Models

No.	Step	Validation Condition
1	Identify the monitored and controlled variables. Describe each of the monitored and controlled variables by its type, its initial value, its accuracy, and a comment.	-
2	Identify constraints imposed by the environment on the monitored and controlled variables. Specify the environmental assertions.	-
3	For each controlled variable, describe the required relation between monitored variables and the controlled variable. That is, describe the current value of controlled variable in terms of the current and past values of the monitored variables. Partition a complex function into a composition of sub-functions, if necessary, by introducing terms and mode classes. Define mode classes by mode transition tables. Define terms and controlled variables by condition tables or event tables.	Table Entry, Cycle Detection
3.1	*Mode transition tables*: Identify and define the modes of the mode class. Identify the events of interest, which shall be processed by this mode machine. Describe the possible mode transitions.	Disjointness, Mode Reachability, Type
3.2	*Event tables*: Identify the events of interest. Identify the desired changes of the term or controlled variable. Describe the possible transitions from one value to the next.	Type, Disjointness
3.3	*Condition tables*: Identify all entities that affect the term or controlled variable. State conditions that partition the input domain into regions that map to a common value in the output range. Identify the value of the term or controlled variable for each condition.	Type, Coverage, Disjointness, Consistent Initial State
4	Describe the invariants of the requirements model by specification assertions.	Prove Assertions

The steps of the agenda are briefly explained. The description intended to be self-contained; therefore, some information given before is repeated.

Step 1: Monitored variables represent environmental quantities that affect the system's behavior and controlled variables represent environmental quantities

that are affected by the system's behavior. Examples of monitored variables include sensors such as temperature and pressure sensors. Examples of controlled variables include actuators such as valves and heating devices. Monitored and controlled variables establish the boundary between the environment and the system. Therefore, they must be directly measurable by the system[1]. Use a comment to describe the associated device. If a variable of interest is not directly measurable[2], use a comment to describe the assumptions made about properties of the environment in order to conclude from the measured variables the value oft the variable of interest.

Step 2: There may be values in the range of a monitored variable, which never naturally occur because of properties of the environment. Similarly, there may be sequences of events that never occur. Occasionally, particular values of the controlled variables prevent the occurrence of some values or events of the monitored variables. All these cases are documented as environmental assumptions. The system is required to deal only with the cases that are not excluded by the environmental assertions.[3]

Step 3: A controlled variable function should be partitioned into a composition of smaller functions, if (1) different controlled variable functions have commonalities, (2) this partition exists already in the problem domain, or (3) the function would be too complex to understand otherwise. Partitioning thus makes the requirements model more concise.

Decide whether the history of events plays a role, that is, whether the occurrence of particular events affect future system behavior. If the history plays a role, introduce a mode class, when there is a limited number of distinguishable classes of event histories for which you can find meaningful names. Otherwise, introduce a term.

Define terms and controlled variables by condition tables, if they are affected only by the current values of other entities. Define them by an event table, if they are affected also by the past values of itself or other entities. Use condition tables as much as possible, because they allow for more consistency checks.[4]

Step 4: The invariants of the requirements model should be stated as specification assertions, because they can then be tested by a model-checker or during the simulation of the requirements model. The model checker tests the complete state space of the requirements model, within certain limits, while simulation is necessarily incomplete.

1 They are *shared phenomena* in Jackson's terminology [Jac95].
2 They are *unshared phenomena* in Jackson's terminology [Jac95].
3 This step aims to define the NAT relation [PM95] or indicative statements [Jac95].
4 This step aims to define the REQ relation [PM95] or optative statements [Jac95].

Anhang B Fragen zur textuellen TSG-Spezifikation

Im folgenden werden Fehler bzw. Unklarheiten in der TSG-Beschreibung in einem Frage-Antwort-Stil dargestellt. Die Antworten, welche von den Autoren dieses Berichts gegeben wurden, bilden Voraussetzungen für die Modellierung. Angegebene Seitenzahlen beziehen sich auf die Spezifikation des TSGs.

1. Frage:
   Kann Taster zur Sitzsteuerung direkt von heben auf senken gestellt werden, oder liegt immer ein ruhen-Zustand dazwischen?

   Antwort:
   Ein Taster kann nicht von Heben auf Senken gestellt werden (Annahme bei Modellierung, da wir uns nicht vorstellen konnten wie es anders gehen soll.)

2. Frage:
   Ist Geschwindigkeitsbegrenzung (<5 km/h) für den manuellen Fall der Sitzsteuerung wichtig?

   Antwort:
   Nein, nur abhängig von Tür auf/zu, da keiner mit geöffneter Tür losfährt und den Sitz dann noch verstellt (normalerweise)!

3. Frage:
   Warum werden Can.LL und Can.RL nicht in Konfigurationsspeicher des TSGs gespeichert? Sie verändern sich doch während des Betriebs des TSGs nicht!

   Antwort:
   Wir modellieren Can.LL und Can.RL als wären Sie im Konfigurationsspeicher abgelegt.

4. Frage:
   Warum gibt es für die verschiedenen Motoren jeweils 2 Leitungen zur Ansteuerung (z.B. S2.Smot_h1 und h2) wenn auf sie jeweils die gleiche Spannung gegeben wird?

   Antwort;
   Unklar, wir modellieren nur eine (die dann ja über Y- Kabel zur Ansteuerung eines 2. Motors gesplittet werden könnte).

5. Frage:
   Was soll passieren, wenn 3 Tasten gedrückt werden, 2 Motoren anspringen und dann eine der Tasten losgelassen wird, deren Motor läuft? Soll der Motor der 3. Taste nun loslaufen oder weiter ausgeschaltet bleiben?

   Antwort;
   Man drückt Taste A,B und C. Es bewegen sich MotA und MotB, MotC bleibt in Ruhen (da nach Spezifikation nur 2 Motoren gleichzeitig laufen dürfen). Nun wird B losgelassen. MotB stoppt. Soll MotC nun anspringen? Wir lassen MotC aus. Die Taste C muß losgelassen und erneut gedrückt werden. Es gäbe sonst eine Art überraschungseffekt.

6. Frage:
   Wie ist Recover, wenn Batteriespannung kleiner als Schwellwert war?

   Antwort:
   Auch wenn noch Taster gedrückt sind, muß der Benutzer diese loslassen und erneut drücken, damit die Motoren wieder arbeiten.

7. Frage:
   Seite 18 Abb. 9 und Seite 19: Warum tiefer und höher? Inkonsistent mit Seite 19: heben und senken.

   Antwort:
   Für die Modellierung bleiben wir bei heben und senken.

8. Frage:
   Seite 19, Tabelle 3: Warum ist Bewegung bei Schalung heben und senken, und nicht enger und weiter?

   Antwort:
   Bewegung der Schalung wird in der Modellierung enger und weiter genannt.

9. Frage:
   Seite 32: Wird die Sitzsteuerung über das Benutzermanagement schon abgebrochen wenn FZG_V (Fahrzeuggeschwindigkeit) gleich 5 km/h oder erst wenn sie größer 5 km/h ist?

   Antwort:
   Die Bewegung wird abgebrochen, wenn FZG_V > 5 km/h.

10. Frage:
Seite 34: Was bedeutet gleichzeitiges Drücken zweier Tasten (MGMT_1, MGMT_Set) zum Speichern von Sitzpositionen? (In SCR kann zeitgleiches Drücken nicht modelliert werden.)

Antwort:
In SCR können 2 Events, die durch 2 verschiedene Variablen ausgelöst werden nicht zeitgleich eintreten.Wir modellieren das Verhalten so, daß zuerst der Taster MGMT_Set gehalten werden muß und dann eine Speicherposition (z.B. MGMT_1)gedrückt wird.

11. Frage:
Seite 34: Es wird nicht klar, ob die Sitzeinstellung über Benutzermanagement auch für den Beifahrersitz möglich ist.

Antwort:
Wir nehmen an, daß die Sitzposition des Beifahrersitzes nicht mitabgespeichert wird.

12. Frage:
Seite 32: Was passiert wenn Sitzansteuerung über den Funksender aktiv ist und der Benutzer drückt einen Taster?

Antwort:
Der Tastendruck hat höhere Priorität, die Bewegung über den Funksender wird abgebrochen.(Wir haben uns hier an der Anforderungsbeschreibung für die Prioritätenregelung beim Benutzermanagement (Seite 32) angelehnt.)

13. Frage:
Kann der Funksender genutzt werden, wenn das Fahrzeug in Bewegung ist (FZG_V >0)?

Antwort:
Nein, nur bei FZG_V = 0.

14. Frage:
Wie ist das Verhalten, wenn Sitzposition durch Benutzermanagement-taster verändert wird und nun Funksender betätigt wird.

Antwort:
Der Funksender hat höhere Priorität.

15.Frage:
Was passiert, wenn Speicherposition 1 durch den Funksender abgerufen wird und während der Ausführung Speicherposition 2 durch den Funksender abgerufen wird?

Antwort:
Der Sitz wird entsprechend Sitzposition 2 angesteuert.

16.Frage:
Seite 24: Warum sind M_Pos_X und M_Save_X zyklische Botschaften?

Antwort:
Wir haben sie als einmaliges Event modelliert.

17.Frage:
Hat das TSG einen internen Speicher, oder wie werden sonst Speicherpositionen übertragen (z.B. Can)?

Antwort:
Das TSG hat einen internen Speicher.

18.Frage:
Seite 26: Konfigurierbarkeit: Was bedeutet aktiv bzw. aktiviert? (Unterschied)

Antwort: ?

19.Frage:
Seite 34: Was bedeutet "Intern gibt es Verbindungen zur Sitzeinstellung"?

Antwort: ?

20.Frage:
Wenn über die Benutzermanagementtaster eine Sitzposition abgerufen wird, dürfen sich dann auch 2 Motoren gleichzeitig bewegen, obwohl ein Motor zur Entspannung der Sitzposition führt und einer zu einer Einschränkung?

Antwort:
Wir nehmen an, daß erst alle entspannenden Bewegungen ausgeführt werden sollen und dann die einschränkenden. Dies steht so in der Spezifikation wenn man sie wörtlich nimmt. Allerdings könnte man durch das gleichzeitige Ansteuern beider Motoren ein wenig Zeit gewinnen.

Anhang C Modell des Türsteuergerätes

Im folgenden ist das Modell des TSG, d.h. der Sitzsteuerung, in SCR zu finden. Diese Spezifikation wurde aus dem SCR Toolset generiert.

Table 2: Type Dictionary

Name	Base Type	Units	Legal Values	Comment
Position_TSG	Enumerated		nicht_gesetzt,links, rechts,ungueltig	S.26
Sitz_motor_horizontal	Enumerated		nach_vorne,ruhen, nach_hinten	S. 19
Sitz_motor_schalung	Enumerated		enger,ruhen,weiter	S.19
Sitz_motor_vorne_hinten	Enumerated		heben,ruhen,senken	S.19
Sitz_motor_winkel	Enumerated		steiler,ruhen,flacher	S.19
Sitz_position	Integer		[0,10]	S.18 0 bedeutet: Entfernung Lenkrad-Sitz minimal, Schalung am engsten, Sitz in der tiefsten Position, Lehne am weitesten zum Lenkrad geneigt. 10 bedeutet maximale Ausdehnung in die jeweils andere Richtung.
Sitz_taster_horizontal	Enumerated		nach_vorne,ruhen, nach_hinten	S. 17
Sitz_taster_schalung	Enumerated		enger,ruhen,weiter	S.17
Sitz_taster_vorne_hinten	Enumerated		heben,ruhen,senken	S.17
Sitz_taster_winkel	Enumerated		steiler,ruhen,flacher	S.17
Spannung	Integer		[50,180]	S.22 Achtung: Einheit:100mV

Table 3: Mode Class Dictionary

Name	Modes	Initial Mode	Table?	Comment
Sitzeinstellung	Ruhen,Tuer_offen, Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender, BM_Sitz_bewegen _entspannen, BM_Sitz_bewegen _einschraenken	Ruhen	Yes	Uebergeordnete Mode Klasse fuer die Sitzeinstellung. In Ruhen ist die Tuer geschlossen und es laeuft kein Sitzmotor. In Tuer_offen laeuft kein Sitzmotor und die Tuer ist auf. In Sitz_bewegen_1 laeuft 1 Sitzmotor nachdem ein Sitztaster gedrueckt wurde. In Sitz_bewegen_2 laufen 2 Sitzmotoren nachdem mindestens zwei Sitztaster gedrueckt wurden. In den Zustand BM_Funksender geht das TSG, wenn das TSG das Fahrertsg ist und ein entsprechendes Signal vom Funksender empfaengt. In den Zustand BM_Sitz_bewegen_entspannen geht das TSG, wenn der Benutzer einen der zwei Benutzermanagementtaster betaetigt, und es Bewegungen gibt, die zur Entspannung der Sitzposition fuehren. (S.32) In den Zustand BM_Sitz_bewegen_einschraenken geht das TSG, wenn der Benutzer einen der zwei Benutzermanagementtaster betaetigt hat und es keine Bewegungen (mehr) gibt, die zur Entspannung der Sitzposition fuehren. (S.32)
Taktgeber	Move, Elapsed	Move	Yes	Diese Mode Klasse ist zur Modellierung von Zeiteinheiten eingefuehrt.

Table 4: Constant Dictionary (Sheet 1 of 2)

Name	Class	Type	Value	Comment
BATTERIE_SCHWELLWERT	Constant	Spannung	100	S.32 Achtung: Einheit 100mV
CONFIG_SITZ_BF	Constant	Boolean	FALSE	TSG am Beifahrersitz montiert? S.26, S.31
CONFIG_SITZ_F	Constant	Boolean	TRUE	TSG am Fahrersitz montiert? S.26, S.31
CONFIG_TSG_LEFT	Constant	Position_TSG	links	TSG auf der linken Seite des Fahrzeugs montiert? S.26, S.31
IN_LL	Constant	Boolean	TRUE	Linkslenker-Fahrzeug? S.23, S.31
IN_RL	Constant	Boolean	FALSE	Rechtslenker-Fahrzeug? S.23, S.31
MOVEMENT	Constant	Integer	1	Bewegung des Sitzes in normierter Darstellung. Bei Takt wird Position um diesen Wert geaendert, falls Motor aktiv.

Table 4: Constant Dictionary (Sheet 2 of 2)

Name	Class	Type	Value	Comment
MOVING_TIME	Constant	Integer	1	Diese Zeiteinheit wird vom Taktgeber genutzt um die Zeit fortzuschalten.
Maximalwert	Constant	Integer	10	Normierter Maximalwert der verschiedenen Sitzpositionen (Abstand Sitz - Lenkrad, Lehnenwinkel, Schalung, Sitzhoehe vornen bzw hinten)
Minimalwert	Constant	Integer	0	Normierter Minimalwert der verschiedenen Sitzpositionen (Abstand Sitz - Lenkrad, Lehnenwinkel, Schalung, Sitzhoehe vornen bzw hinten)
V_SCHWELLWERT	Constant	Integer	20	Wert, den die Fahrzeuggeschwindigkeit nicht ueberschreiten darf, damit eine Ansteuerung der Sitzmotoren durch das Benutzermanagement noch moelich ist. S. 32

Table 5: Monitored Variable Dictionary (Sheet 1 of 4)

Name	Type	Initial Value	Accuracy	Comment
mBatt	Spannung	120		Gibt die Batteriespannung des Fahrzeugs an. S. 22
mBlockiert_h	Boolean	FALSE		Hier wird die Blockierung eines Motors simuliert. Dies ist ueber eine monitorierte Variable noetig, da die Sitzposition im Environment blockiert werden wuerde. S. 33
mBlockiert_hor	Boolean	FALSE		Hier wird die Blockierung eines Motors simuliert. Dies ist ueber eine monitorierte Variable noetig, da die Sitzposition im Environment blockiert werden wuerde. S. 33
mBlockiert_s	Boolean	FALSE		Hier wird die Blockierung eines Motors simuliert. Dies ist ueber eine monitorierte Variable noetig, da die Sitzposition im Environment blockiert werden wuerde. S. 33

Table 5: Monitored Variable Dictionary (Sheet 2 of 4)

Name	Type	Initial Value	Accuracy	Comment
mBlockiert_v	Boolean	FALSE		Hier wird die Blockierung eines Motors simuliert. Dies ist ueber eine monitorierte Variable noetig, da die Sitzposition im Environment blockiert werden wuerde. S. 33
mBlockiert_w	Boolean	FALSE		Hier wird die Blockierung eines Motors simuliert. Dies ist ueber eine monitorierte Variable noetig, da die Sitzposition im Environment blockiert werden wuerde. S. 33
mFunksender_Pos 1	Boolean	FALSE		Simulation der Funksendertaste des 1. Benutzers. S. 9, 25
mFunksender_Pos 2	Boolean	FALSE		Simulation der Funksendertaste des 2. Benutzers. S. 9, 25
mIn_Fzg_v	Integer	0		Aktuelle Fahrzeuggeschwindigkeit. S.23

Table 5: Monitored Variable Dictionary (Sheet 3 of 4)

Name	Type	Initial Value	Accuracy	Comment
mMgmt_1	Boolean	FALSE		Variable entspricht dem Taster zum Abrufen einer abgespeicherten Speicherposition. In Kombination mit der gehaltenen mMgmt_set-Taste dient sie auch zum Abspeichern der aktuellen Sitzposition unter Speicherposition 1. S.13
mMgmt_2	Boolean	FALSE		Variable entspricht dem Taster zum Abrufen einer abgespeicherten Speicherposition. In Kombination mit der gehaltenen mMgmt_set-Taste dient sie auch zum Abspeichern der aktuellen Sitzposition unter Speicherposition 2. S.13
mMgmt_set	Boolean	FALSE		Taster zum Abspeichern der Speicherposition. Wird sie gehalten und die Taste mMgmt_1 oder mMgmt_2 gedrueckt, so wird die aktuelle Sitzposition unter Speicherposition 1 bzw. 2 gespeichert. S.13

Table 5: Monitored Variable Dictionary (Sheet 4 of 4)

Name	Type	Initial Value	Accuracy	Comment
mSitz_h	Sitz_taster_vorne_hinten	ruhen		Variable entspricht dem Taster um den Sitzmotor zum hinten heben bzw. senken anzus-teuern. S.16, S.31
mSitz_hor	Sitz_taster_horizontal	ruhen		Variable entspricht dem Taster um den horizontalen Sitz-motor direkt anzusteuern. S.16, S.31
mSitz_s	Sitz_taster_schalung	ruhen		Variable entspricht dem Taster um den Sitzmotor fuer die Schalung direkt anzusteuern. S.16, S.31
mSitz_v	Sitz_taster_vorne_hinten	ruhen		Variable entspricht dem Taster um den Sitzmotor zum vorne heben bzw. senken anzus-teuern. S.16, S.31
mSitz_w	Sitz_taster_winkel	ruhen		Variable entspricht dem Taster um den Sitzmotor fuer die Lehne (Winkel) direkt anzus-teuern. S.16, S.31
mT_offen	Boolean	FALSE		Gibt an, ob die Tuer des Autos auf der Seite des TSGs geoeffnet ist. S.13, S.31
time	Integer	0		Globale Zeit

Table 6: Term Dictionary (Sheet 1 of 5)

Name	Type	Initial Value	Accuracy	Comment
tAktuelle_Speicher_Position	Integer	0		Term, der die ausgewaehlte Speicherposition enthaelt (1 oder 2), wenn mit dem Funksender oder einem Managementtaster eine Benutzereinstellung gewaehlt wurde. Term wird 0 sobald die Speicherposition erreicht wird.
tBM_Blockierung_aufgetreten_h	Boolean	FALSE		Prueft, ob beim Ansteuern der Motoren initiiert durch ein Funksendersignal eine Blockierung aufgetreten ist.
tBM_Blockierung_aufgetreten_hor	Boolean	FALSE		Prueft, ob beim Ansteuern der Motoren initiiert durch ein Funksendersignal eine Blockierung aufgetreten ist.
tBM_Blockierung_aufgetreten_s	Boolean	FALSE		Prueft, ob beim Ansteuern der Motoren initiiert durch ein Funksendersignal eine Blockierung aufgetreten ist.

Table 6: Term Dictionary (Sheet 2 of 5)

Name	Type	Initial Value	Accuracy	Comment
tBM_Blockierung_aufgetreten_v	Boolean	FALSE		Prueft, ob beim Ansteuern der Motoren initiiert durch ein Funksendersignal eine Blockierung aufgetreten ist.
tBM_Blockierung_aufgetreten_w	Boolean	FALSE		Prueft, ob beim Ansteuern der Motoren initiiert durch ein Funksendersignal eine Blockierung aufgetreten ist.
tBM_einschraenken_h	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Einschraenkung der Sitzposition angesteuert werden soll und darf. S. 32
tBM_einschraenken_hor	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Einschraenkung der Sitzposition angesteuert werden soll und darf. S. 32
tBM_einschraenken_s	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Einschraenkung der Sitzposition angesteuert werden soll und darf. S. 32

Table 6: Term Dictionary (Sheet 3 of 5)

Name	Type	Initial Value	Accuracy	Comment
tBM_einschraenken_v	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Einschraenkung der Sitzposition angesteuert werden soll und darf. S. 32
tBM_einschraenken_w	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Einschraenkung der Sitzposition angesteuert werden soll und darf. S. 32
tBM_entspannen_h	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Entspannung der Sitzposition angesteuert werden soll und darf. S. 32
tBM_entspannen_hor	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Entspannung der Sitzposition angesteuert werden soll und darf. S. 32
tBM_entspannen_s	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Entspannung der Sitzposition angesteuert werden soll und darf. S. 32

Table 6: Term Dictionary (Sheet 4 of 5)

Name	Type	Initial Value	Accuracy	Comment
tBM_entspannen_v	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Entspannung der Sitzposition angesteuert werden soll und darf. S. 32
tBM_entspannen_w	Boolean	FALSE		Term, der anzeigt, ob der entsprechende Motor zu einer Entspannung der Sitzposition angesteuert werden soll und darf. S. 32
tHinten_bewegen	Boolean	FALSE		Term wird zur besseren Lesbarkeit eingefuehrt, er sagt aus, ob sich der Motor bewegen soll oder nicht.
tHor_bewegen	Boolean	FALSE		Term wird zur besseren Lesbarkeit eingefuehrt, er sagt aus, ob sich der Motor bewegen soll oder nicht.
tSchalung_bewegen	Boolean	FALSE		Term wird zur besseren Lesbarkeit eingefuehrt, er sagt aus, ob sich der Motor bewegen soll oder nicht.

Table 6: Term Dictionary (Sheet 5 of 5)

Name	Type	Initial Value	Accuracy	Comment
tSitzeinstellung_aktiviert	Boolean	TRUE		Term wird zur besseren Lesbarkeit eingefuehrt, er gibt an ob im TSG die Sitzeinstellung aktiviert ist.
tSpeicher_Position 1_abgleich	Boolean	FALSE		Abgleich der Speicherposition 1
tSpeicher_Position 2_abgleich	Boolean	FALSE		Abgleich der Speicherposition 2
tVorne_bewegen	Boolean	FALSE		Term wird zur besseren Lesbarkeit eingefuehrt, er sagt aus, ob sich der Motor bewegen soll oder nicht.
tWinkel_bewegen	Boolean	FALSE		Term wird zur besseren Lesbarkeit eingefuehrt, er sagt aus, ob sich der Motor bewegen soll oder nicht.

Table 7: Controlled Variable Dictionary (Sheet 1 of 4)

Name	Type	Initial Value	Accuracy	Comment
cFehler_sitz_blockiert	Boolean	FALSE		Simuliert Eintrag in Fehlerspeicher. S. 33
cIn_M_pos_1	Boolean	FALSE		Aufruf zum Abruf der Speicherposition 1 von Funksendereinheit. S. 24
cIn_M_pos_2	Boolean	FALSE		Aufruf zum Abruf der Speicherposition 2 von Funksendereinheit. S. 24
cOut_b_low_sitz	Boolean	FALSE		Nachricht aus dem Tsgsystem hinaus an andere Komponenten im Fahrzeug. S.24, S.31
cSmot_h	Sitz_motor_vorne_hinten	ruhen		Aktuelle Ansteuerung des Sitzmotors um ihn hinten zu heben bzw. zu senken. S.17, S.31
cSmot_hor	Sitz_motor_horizontal	ruhen		Aktuelle Ansteuerung des Sitzmotors zur horizontalen Bewegung. S.16, S.31
cSmot_s	Sitz_motor_schalung	ruhen		Aktuelle Ansteuerung des Sitzmotors zum Einstellen der Schalung. S.17, S.31

Table 7: Controlled Variable Dictionary (Sheet 2 of 4)

Name	Type	Initial Value	Accuracy	Comment
cSmot_v	Sitz_motor_vorne_hinten	ruhen		Aktuelle Ansteuerung des Sitzmotors um ihn vorne zu heben bzw. zu senken.S.16, S.31
cSmot_w	Sitz_motor_winkel	ruhen		Aktuelle Ansteuerung des Sitzmotors zum Einstellen des Winkels der Lehne. S.17, S.31
cSpeicher_Pos1_h	Sitz_position	2		Speicherposition 1 fuer Sitzeinstellung hinten. S. 34
cSpeicher_Pos1_hor	Sitz_position	9		Speicherposition 1 fuer horizontale Sitzeinstellung. S. 34
cSpeicher_Pos1_s	Sitz_position	10		Speicherposition 1 fuer Sitzeinstellung Schalung. S. 34
cSpeicher_Pos1_v	Sitz_position	2		Speicherposition 1 fuer Sitzeinstellung vorne. S. 34
cSpeicher_Pos1_w	Sitz_position	6		Speicherposition 1 fuer Sitzeinstellung Winkel (Lehne). S. 34
cSpeicher_Pos2_h	Sitz_position	2		Speicherposition 2 fuer Sitzeinstellung hinten. S. 34
cSpeicher_Pos2_hor	Sitz_position	3		Speicherposition 2 fuer horizontale Sitzeinstellung. S. 34

Table 7: Controlled Variable Dictionary (Sheet 3 of 4)

Name	Type	Initial Value	Accuracy	Comment
cSpeicher_Pos2_s	Sitz_position	9		Speicherposition 2 fuer Sitzeinstel-lung Schalung. S. 34
cSpeicher_Pos2_v	Sitz_position	9		Speicherposition 2 fuer Sitzeinstel-lung vorne. S. 34
cSpeicher_Pos2_w	Sitz_position	2		Speicherposition 2 fuer Sitzeinstel-lung Winkel (Lehne). S. 34
cSpos_h	Sitz_position	5		Aktuelle Sitzposi-tion hinten. S.16, S.31
cSpos_hor	Sitz_position	5		Aktuelle Sitzposi-tion horizontal. S.16, S.31
cSpos_s	Sitz_position	5		Aktuelle Sitzposi-tion der Schalung. S.16, S.31
cSpos_v	Sitz_position	5		Aktuelle Sitzposi-tion vorne. S.16, S.31
cSpos_w	Sitz_position	5		Aktuelle Sitzposi-tion der Lehne (Winkel) S.16, S.31
cWerte_gespeicher t_1	Boolean	FALSE		Variable gibt an, ob seit der Auslief-erung des Autos eine Benutzerein-stellung unter der Speicherposition 1 abgespeichert wurde. S. 34

Table 7: Controlled Variable Dictionary (Sheet 4 of 4)

Name	Type	Initial Value	Accuracy	Comment
cWerte_gespeicher t_2	Boolean	FALSE		Variable gibt an, ob seit der Auslieferung des Autos eine Benutzereinstellung unter der Speicherposition 2 abgespeichert wurde. S. 34
cZu_Beifahrertsg_ M_pos_mgmt_1	Boolean	FALSE		Aufruf zum Abruf der Speicherposition 1 von Fahrertsg an Beifahrertsg. S. 24
cZu_Beifahrertsg_ M_pos_mgmt_2	Boolean	FALSE		Aufruf zum Abruf der Speicherposition 1 von Fahrertsg an Beifahrertsg. S. 24
cZu_Beifahrertsg_ M_save_1	Boolean	FALSE		Aufruf zum Speichern in Speicherposition 1 an Beifahrertsg. S. 24
cZu_Beifahrertsg_ M_save_2	Boolean	FALSE		Aufruf zum Speichern in Speicherposition 2 an Beifahrertsg. S. 24

Table 8: Mode Transition Table for Sitzeinstellung (Sheet 1 of 5)

Source Mode	Events	Destination Mode
Ruhen	@T(mT_offen = TRUE) WHEN (tSitzeinstellung_aktiviert = TRUE)	Tuer_offen
Tuer_offen	@T(tHinten_bewegen = TRUE) OR @T(tVorne_bewegen = TRUE) OR @T(tHor_bewegen = TRUE) OR @T(tSchalung_bewegen = TRUE) OR @T(tWinkel_bewegen = TRUE)	Sitz_bewegen_1
Sitz_bewegen_1	@T(tHinten_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tVorne_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tHor_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tSchalung_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tWinkel_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)'	Tuer_offen
Sitz_bewegen_1	@T(tHinten_bewegen = TRUE) OR @T(tVorne_bewegen = TRUE) OR @T(tHor_bewegen = TRUE) OR @T(tSchalung_bewegen = TRUE) OR @T(tWinkel_bewegen = TRUE)	Sitz_bewegen_2

Table 8: Mode Transition Table for Sitzeinstellung (Sheet 2 of 5)

Source Mode	Events	Destination Mode
Sitz_bewegen_2	@T(tHinten_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tVorne_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tHor_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tSchalung_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)' OR @T(tWinkel_bewegen = FALSE) AND (mT_offen = TRUE) AND (mT_offen = TRUE)' AND (mBatt >= BATTERIE_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)'	Sitz_bewegen_1
Sitz_bewegen_1,Sitz_bewegen_2	@T(mBatt < BATTERIE_SCHWELLWERT)	Tuer_offen
Tuer_offen, Sitz_bewegen_1, Sitz_bewegen_2	@T(mT_offen = FALSE)	Ruhen
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen	@T(cIn_M_pos_1 = TRUE) WHEN ((tSpeicher_Position1_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((tSpeicher_Position2_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT) AND (cWerte_gespeichert_2 = TRUE))	BM_Funksender

Table 8: Mode Transition Table for Sitzeinstellung (Sheet 3 of 5)

Source Mode	Events	Destination Mode
BM_Funksender	@T(tSpeicher_Position1_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 1 AND mT_offen = FALSE) OR @T(tSpeicher_Position2_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 2 AND mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT) WHEN (mT_offen = FALSE) OR @T(mIn_Fzg_v > 0) WHEN (mT_offen = FALSE)	Ruhen
BM_Funksender	@T(tSpeicher_Position1_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 1 AND mT_offen = TRUE) OR @T(tSpeicher_Position2_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 2 AND mT_offen = TRUE) OR @T(mBatt < BATTERIE_SCHWELLWERT) WHEN (mT_offen = TRUE) OR @T(mIn_Fzg_v > 0) WHEN (mT_offen = TRUE)	Tuer_offen
Tuer_offen, Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender, BM_Sitz_bewegen_einschraenken	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND (tBM_entspannen_h =TRUE OR tBM_entspannen_hor =TRUE OR tBM_entspannen_w =TRUE OR tBM_entspannen_v =TRUE OR tBM_entspannen_s =TRUE)' AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)	BM_Sitz_bewegen_entspannen

Table 8: Mode Transition Table for Sitzeinstellung (Sheet 4 of 5)

Source Mode	Events	Destination Mode
Tuer_offen, Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND (tBM_einschraenken_h =TRUE OR tBM_einschraenken_hor =TRUE OR tBM_einschraenken_w =TRUE OR tBM_einschraenken_v =TRUE OR tBM_einschraenken_s =TRUE)' AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT)	BM_Sitz_bewegen_einschraenken
BM_Sitz_bewegen_entspannen	@F(tBM_entspannen_h =TRUE OR tBM_entspannen_hor =TRUE OR tBM_entspannen_w =TRUE OR tBM_entspannen_v =TRUE OR tBM_entspannen_s =TRUE) AND (((tSpeicher_Position1_abgleich = FALSE)' AND tAktuelle_Speicher_Position = 1) OR ((tSpeicher_Position2_abgleich = FALSE)' AND tAktuelle_Speicher_Position = 2)) AND (mIn_Fzg_v <= V_SCHWELLWERT)'	BM_Sitz_bewegen_einschraenken

Table 8: Mode Transition Table for Sitzeinstellung (Sheet 5 of 5)

Source Mode	Events	Destination Mode
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(tSpeicher_Position1_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 1 AND mT_offen = FALSE) OR @T(tSpeicher_Position2_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 2 AND mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT) WHEN (mT_offen = FALSE) OR @T(mIn_Fzg_v > V_SCHWELLWERT) WHEN (mT_offen = FALSE)	Ruhen
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(tSpeicher_Position1_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 1 AND mT_offen = TRUE) OR @T(tSpeicher_Position2_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 2 AND mT_offen = TRUE) OR @T(mBatt < BATTERIE_SCHWELLWERT) WHEN (mT_offen = TRUE) OR @T(mIn_Fzg_v > V_SCHWELLWERT) WHEN (mT_offen = TRUE)	Tuer_offen
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	@T(tHinten_bewegen = TRUE) OR @T(tVorne_bewegen = TRUE) OR @T(tHor_bewegen = TRUE) OR @T(tSchalung_bewegen = TRUE) OR @T(tWinkel_bewegen = TRUE)	Sitz_bewegen_1

Notes

Normalerweise werden zur Aenderung der Sitz_bewegen Zust‰nde die MotorTerme herangezogen. Ein Problem gibt es bei Ausnahmef‰llen (Tuer schliessen, Batterie zu schwach). Hier muss direkt in den Zustand Ruhen, bzw. Tuer_offen gegangen werden.

Table 9: Mode Transition Table for Taktgeber

Source Mode	Events	Destination Mode
Move	@T(Duration2(Inmode) > MOVING_TIME)	Elapsed
Elapsed	@C(time)	Move

Table 10: Event Table for cFehler_sitz_blockiert

Name	Mode Class
cFehler_sitz_blockiert	Sitzeinstellung
Modes	**Events**
Sitz_bewegen_1,Sitz_bewegen_2,BM_Funksender, BM_Sitz_bewegen_entspannen,BM_Sitz_bewegen _einschraenken	((@T(Taktgeber = Elapsed) WHEN (cSmot_h!=ruhen)) AND (NOT @C(cSpos_h) WHEN (cSmot_h!=ruhen))) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_v!=ruhen)) AND (NOT @C(cSpos_v) WHEN (cSmot_v!=ruhen))) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_hor!=ruhen)) AND (NOT @C(cSpos_hor) WHEN (cSmot_hor!=ruhen))) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_s!=ruhen)) AND (NOT @C(cSpos_s) WHEN (cSmot_s!=ruhen))) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_w!=ruhen)) AND (NOT @C(cSpos_w) WHEN (cSmot_w!=ruhen)))
cFehler_sitz_blockiert' =	TRUE

Table 11: Event Table for cIn_M_pos_1

Name		
cIn_M_pos_1		
	Events	
	@T(mFunksender_Pos1 = TRUE)	@T(mFunksender_Pos1 = FALSE)
cIn_M_pos_1' =	TRUE	FALSE

Table 12: Event Table for cIn_M_pos_2

Name		
cIn_M_pos_2		
	Events	
	@T(mFunksender_Pos2 = TRUE)	@T(mFunksender_Pos2 = FALSE)
cIn_M_pos_2' =	TRUE	FALSE

Table 13: Event Table for cOut_b_low_sitz

Name		
cOut_b_low_sitz		
	Events	
	@T(mBatt < BATTERIE_SCHWELLWERT)	@T(mBatt >= BATTERIE_SCHWELLWERT)
cOut_b_low_sitz' =	TRUE	FALSE

Table 14: Event Table for cSmot_h (Sheet 1 of 4)

Name		Mode Class	
cSmot_h		Sitzeinstellung	
Modes	**Events**		
Tuer_offen	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = heben)')	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = senken)')	Never
Sitz_bewegen_1	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = heben)')	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = senken)')	@T(tHinten_bewegen = FALSE)
Sitz_bewegen_2	Never	Never	@T(tHinten_bewegen = FALSE)
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_h<cSpeicher_Pos1_h) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_h<cSpeicher_Pos2_h)AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_h>cSpeicher_Pos1_h)AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_h>cSpeicher_Pos2_h)AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	@T(cIn_M_pos_1 = TRUE) WHEN (cSpos_h=cSpeicher_Pos1_h) OR @T(cIn_M_pos_2 = TRUE) WHEN (cSpos_h=cSpeicher_Pos2_h)

Table 14: Event Table for cSmot_h (Sheet 2 of 4)

Name			Mode Class
cSmot_h			Sitzeinstellung
Modes	**Events**		
BM_Funksender	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_h<cSpeicher_Pos1_h)AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_h<cSpeicher_Pos2_h) AND (cWerte_gespeichert_2 = TRUE)))	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_h>cSpeicher_Pos1_h)AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_h>cSpeicher_Pos2_h)AND (cWerte_gespeichert_2 = TRUE)))	@T(cSpos_h=cSpeicher_Pos1_h) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_h=cSpeicher_Pos2_h) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_h= Maximalwert) OR @T(cSpos_h= Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_h!=ruhen)) AND (NOT @C(cSpos_h) WHEN (cSmot_h!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > 0)

Table 14: Event Table for cSmot_h (Sheet 3 of 4)

Name		Mode Class	
cSmot_h		Sitzeinstellung	
Modes	**Events**		
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_einschraenken_h = TRUE)	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_entspannen_h = TRUE)	(@T(mMgmt_1 = TRUE) OR @T(mMgmt_2 = TRUE)) AND (tBM_entspannen_h = FALSE)' AND (tBM_einschraenken_h = FALSE)'
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(tBM_einschraenken_h = TRUE)	@T(tBM_entspannen_h = TRUE)	@T(cSpos_h=cSpeicher_Pos1_h) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_h=cSpeicher_Pos2_h) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_h=Maximalwert) OR @T(cSpos_h=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_h!=ruhen)) AND (NOT @C(cSpos_h) WHEN (cSmot_h!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > V_SCHWELLWERT)

Table 14: Event Table for cSmot_h (Sheet 4 of 4)

Name		Mode Class	
cSmot_h		Sitzeinstellung	
Modes	**Events**		
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	(@T(tHinten_bewegen = TRUE) AND (mSitz_h = heben)')	(@T(tHinten_bewegen = TRUE) AND (mSitz_h=senken)')	@T(tHor_bewegen = TRUE OR tSchalung_bewegen = TRUE OR tVorne_bewegen = TRUE OR tWinkel_bewegen = TRUE)
cSmot_h' =	heben	senken	ruhen

Table 15: Event Table for cSmot_hor (Sheet 1 of 3)

Name		Mode Class	
cSmot_hor		Sitzeinstellung	
Modes	**Events**		
Tuer_offen	@T(tHor_bewegen = TRUE) AND (mSitz_hor = nach_hinten)'	@T(tHor_bewegen = TRUE) AND (mSitz_hor = nach_vorne)'	Never
Sitz_bewegen_1	@T(tHor_bewegen = TRUE) AND (mSitz_hor = nach_hinten)'	@T(tHor_bewegen = TRUE) AND (mSitz_hor = nach_vorne)'	@T(tHor_bewegen = FALSE)
Sitz_bewegen_2	Never	Never	@T(tHor_bewegen = FALSE)
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_hor<cSpeicher_Pos1_hor) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_hor<cSpeicher_Pos2_hor) AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_hor>cSpeicher_Pos1_hor) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_hor>cSpeicher_Pos2_hor) AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	@T(cIn_M_pos_1 = TRUE) WHEN (cSpos_hor=cSpeicher_Pos1_hor) OR @T(cIn_M_pos_2 = TRUE) WHEN (cSpos_hor=cSpeicher_Pos2_hor)

Table 15: Event Table for cSmot_hor (Sheet 2 of 3)

Name		Mode Class	
cSmot_hor		Sitzeinstellung	
Modes	**Events**		
BM_Funksender	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_hor<cSpeicher_Pos1_hor) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_hor<cSpeicher_Pos2_hor) AND (cWerte_gespeichert_2 = TRUE)))	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_hor>cSpeicher_Pos1_hor) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_hor>cSpeicher_Pos2_hor) AND (cWerte_gespeichert_2 = TRUE)))	@T(cSpos_hor=cSpeicher_Pos1_hor) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_hor=cSpeicher_Pos2_hor) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_hor=Maximalwert) OR @T(cSpos_hor=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_hor!=ruhen)) AND (NOT @C(cSpos_hor) WHEN (cSmot_hor!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > 0)

Table 15: Event Table for cSmot_hor (Sheet 3 of 3)

Name		Mode Class	
cSmot_hor		Sitzeinstellung	
Modes	**Events**		
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_entspannen_hor = TRUE)	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_einschraenken_hor = TRUE)	(@T(mMgmt_1 = TRUE) OR @T(mMgmt_2 = TRUE)) AND (tBM_entspannen_hor = FALSE)' AND (tBM_einschraenken_hor = FALSE)'
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(tBM_entspannen_hor = TRUE)	@T(tBM_einschraenken_hor = TRUE)	@T(cSpos_hor=cSpeicher_Pos1_hor) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_hor=cSpeicher_Pos2_hor) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_hor=Maximalwert) OR @T(cSpos_hor=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_hor!=ruhen)) AND (NOT @C(cSpos_hor) WHEN (cSmot_hor!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > V_SCHWELLWERT)
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	@T(tHor_bewegen = TRUE) AND (mSitz_hor = nach_hinten)'	@T(tHor_bewegen = TRUE) AND (mSitz_hor = nach_vorne)'	@T(tHinten_bewegen = TRUE OR tSchalung_bewegen = TRUE OR tVorne_bewegen = TRUE OR tWinkel_bewegen = TRUE)
cSmot_hor' =	nach_hinten	nach_vorne	ruhen

Table 16: Event Table for cSmot_s (Sheet 1 of 3)

Name		Mode Class	
cSmot_s		Sitzeinstellung	
Modes	**Events**		
Tuer_offen	@T(tSchalung_bewegen = TRUE) AND (mSitz_s=weiter)'	@T(tSchalung_bewegen = TRUE) AND (mSitz_s=enger)'	Never
Sitz_bewegen_1	@T(tSchalung_bewegen = TRUE) AND (mSitz_s=weiter)'	@T(tSchalung_bewegen = TRUE) AND (mSitz_s=enger)'	@T(tSchalung_bewegen = FALSE)
Sitz_bewegen_2	Never	Never	@T(tSchalung_bewegen = FALSE)
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_s<cSpeicher_Pos1_s) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_s<cSpeicher_Pos2_s) AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_s>cSpeicher_Pos1_s) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_s>cSpeicher_Pos2_s)AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	@T(cIn_M_pos_1 = TRUE) WHEN (cSpos_s=cSpeicher_Pos1_s) OR @T(cIn_M_pos_2 = TRUE) WHEN (cSpos_s=cSpeicher_Pos2_s)

Table 16: Event Table for cSmot_s (Sheet 2 of 3)

Name			Mode Class
cSmot_s			Sitzeinstellung
Modes	**Events**		
BM_Funksender	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_s<cSpeicher_Pos 1_s) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_s<cSpeicher_Pos 2_s)AND (cWerte_gespeichert_2 = TRUE)))	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_s>cSpeicher_Pos 1_s) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_s>cSpeicher_Pos 2_s)AND (cWerte_gespeichert_2 = TRUE)))	@T(cSpos_s=cSpeicher_Pos 1_s) WHEN (tAktuelle_Speicher_Positio n = 1) OR @T(cSpos_s=cSpeicher_Pos 2_s) WHEN (tAktuelle_Speicher_Positio n = 2) OR @T(cSpos_s=Maximalw-ert) OR @T(cSpos_s=Mini-malwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_s!=ruhen)) AND (NOT @C(cSpos_s) WHEN (cSmot_s!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > 0)

Table 16: Event Table for cSmot_s (Sheet 3 of 3)

Name		Mode Class	
cSmot_s		Sitzeinstellung	
Modes	**Events**		
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_entspannen_s = TRUE)	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_einschraenken_s = TRUE)	(@T(mMgmt_1 = TRUE) OR @T(mMgmt_2 = TRUE)) AND (tBM_entspannen_s = FALSE)' AND (tBM_einschraenken_s = FALSE)'
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(tBM_entspannen_s = TRUE)	@T(tBM_einschraenken_s = TRUE)	@T(cSpos_s=cSpeicher_Pos1_s) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_s=cSpeicher_Pos2_s) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_s=Maximalwert) OR @T(cSpos_s=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_s!=ruhen)) AND (NOT @C(cSpos_s) WHEN (cSmot_s!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > V_SCHWELLWERT)
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	@T(tSchalung_bewegen = TRUE) AND (mSitz_s=weiter)'	@T(tSchalung_bewegen = TRUE) AND (mSitz_s=enger)'	@T(tHor_bewegen = TRUE OR tHinten_bewegen = TRUE OR tVorne_bewegen = TRUE OR tWinkel_bewegen = TRUE)
cSmot_s' =	weiter	enger	ruhen

Table 17: Event Table for cSmot_v (Sheet 1 of 3)

Name		Mode Class	
cSmot_v		Sitzeinstellung	
Modes	**Events**		
Tuer_offen	@T(tVorne_bewegen = TRUE) AND (mSitz_v=heben)'	@T(tVorne_bewegen = TRUE) AND (mSitz_v=senken)'	Never
Sitz_bewegen_1	@T(tVorne_bewegen = TRUE) AND (mSitz_v=heben)'	@T(tVorne_bewegen = TRUE) AND (mSitz_v=senken)'	@T(tVorne_bewegen = FALSE)
Sitz_bewegen_2	Never	Never	@T(tVorne_bewegen = FALSE)
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_v<cSpeicher_Pos1_v) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_v<cSpeicher_Pos2_v)AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_v>cSpeicher_Pos1_v) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_v>cSpeicher_Pos2_v) AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	@T(cIn_M_pos_1 = TRUE) WHEN (cSpos_v=cSpeicher_Pos1_v) OR @T(cIn_M_pos_2 = TRUE) WHEN (cSpos_v=cSpeicher_Pos2_v)

Table 17: Event Table for cSmot_v (Sheet 2 of 3)

Name		Mode Class	
cSmot_v		Sitzeinstellung	
Modes	**Events**		
BM_Funksender	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_v<cSpeicher_Pos1_v) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_v<cSpeicher_Pos2_v)AND (cWerte_gespeichert_2 = TRUE)))	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_v>cSpeicher_Pos1_v) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_v>cSpeicher_Pos2_v) AND (cWerte_gespeichert_2 = TRUE)))	@T(cSpos_v=cSpeicher_Pos1_v) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_v=cSpeicher_Pos2_v) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_v=Maximalwert) OR @T(cSpos_v=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_v!=ruhen)) AND (NOT @C(cSpos_v) WHEN (cSmot_v!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > 0)

Table 17: Event Table for cSmot_v (Sheet 3 of 3)

Name		Mode Class	
cSmot_v		Sitzeinstellung	
Modes	**Events**		
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_einschraenken_v = TRUE)	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_entspannen_v = TRUE)	(@T(mMgmt_1 = TRUE) OR @T(mMgmt_2 = TRUE)) AND (tBM_entspannen_v = FALSE)' AND (tBM_einschraenken_v = FALSE)'
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(tBM_einschraenken_v = TRUE)	@T(tBM_entspannen_v = TRUE)	@T(cSpos_v=cSpeicher_Pos1_v) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_v=cSpeicher_Pos2_v) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_v=Maximalwert) OR @T(cSpos_v=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_v!=ruhen)) AND (NOT @C(cSpos_v) WHEN (cSmot_v!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > V_SCHWELLWERT)
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	@T(tVorne_bewegen = TRUE) AND (mSitz_v=heben)'	@T(tVorne_bewegen = TRUE) AND (mSitz_v=senken)'	@T(tHor_bewegen = TRUE OR tSchalung_bewegen = TRUE OR tHinten_bewegen = TRUE OR tWinkel_bewegen = TRUE)
cSmot_v' =	heben	senken	ruhen

Table 18: Event Table for cSmot_w (Sheet 1 of 4)

Name		Mode Class	
cSmot_w		Sitzeinstellung	
Modes	**Events**		
Tuer_offen	@T(tWinkel_bewegen = TRUE) AND (mSitz_w=flacher)'	@T(tWinkel_bewegen = TRUE) AND (mSitz_w=steiler)'	Never
Sitz_bewegen_1	@T(tWinkel_bewegen = TRUE) AND (mSitz_w=flacher)'	@T(tWinkel_bewegen = TRUE) AND (mSitz_w=steiler)'	@T(tWinkel_bewegen = FALSE)
Sitz_bewegen_2	Never	Never	@T(tWinkel_bewegen = FALSE)
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_w<cSpeicher_Pos1_w) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_w<cSpeicher_Pos2_w) AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_w>cSpeicher_Pos1_w) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_w>cSpeicher_Pos2_w)AND (cWerte_gespeichert_2 = TRUE))) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT)	@T(cIn_M_pos_1 = TRUE) WHEN (cSpos_w=cSpeicher_Pos1_w) OR @T(cIn_M_pos_2 = TRUE) WHEN (cSpos_w=cSpeicher_Pos2_w)

Table 18: Event Table for cSmot_w (Sheet 2 of 4)

Name			Mode Class
cSmot_w			Sitzeinstellung
Modes	**Events**		
BM_Funksender	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_w<cSpeicher_Pos1_w) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_w<cSpeicher_Pos2_w)AND (cWerte_gespeichert_2 = TRUE)))	(@T(cIn_M_pos_1 = TRUE) WHEN ((cSpos_w>cSpeicher_Pos1_w) AND (cWerte_gespeichert_1 = TRUE)) OR @T(cIn_M_pos_2 = TRUE) WHEN ((cSpos_w>cSpeicher_Pos2_w)AND (cWerte_gespeichert_2 = TRUE)))	@T(cSpos_w=cSpeicher_Pos1_w) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_w=cSpeicher_Pos2_w) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_w=Maximalwert) OR @T(cSpos_w=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_w!=ruhen)) AND (NOT @C(cSpos_w) WHEN (cSmot_w!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > 0)

Table 18: Event Table for cSmot_w (Sheet 3 of 4)

Name		Mode Class	
cSmot_w		Sitzeinstellung	
Modes	**Events**		
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_entspannen_w = TRUE)	((@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)) OR (@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE))) AND @T(tBM_einschraenken_w = TRUE)	(@T(mMgmt_1 = TRUE) OR @T(mMgmt_2 = TRUE)) AND (tBM_entspannen_w = FALSE)' AND (tBM_einschraenken_w = FALSE)'
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(tBM_entspannen_w = TRUE)	@T(tBM_einschraenken_w = TRUE)	@T(cSpos_w=cSpeicher_Pos1_w) WHEN (tAktuelle_Speicher_Position = 1) OR @T(cSpos_w=cSpeicher_Pos2_w) WHEN (tAktuelle_Speicher_Position = 2) OR @T(cSpos_w=Maximalwert) OR @T(cSpos_w=Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_w!=ruhen)) AND (NOT @C(cSpos_w) WHEN (cSmot_w!=ruhen))) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > V_SCHWELLWERT)
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	@T(tWinkel_bewegen = TRUE) AND (mSitz_w=flacher)'	@T(tWinkel_bewegen = TRUE) AND (mSitz_w=steiler)'	@T(tHinten_bewegen = TRUE OR tSchalung_bewegen = TRUE OR tVorne_bewegen = TRUE OR tHor_bewegen = TRUE)

Table 18: Event Table for cSmot_w (Sheet 4 of 4)

Name		Mode Class	
cSmot_w		Sitzeinstellung	
Modes	**Events**		
cSmot_w' =	flacher	steiler	ruhen

Table 19: Event Table for cSpeicher_Pos1_h

Name	Mode Class
cSpeicher_Pos1_h	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_1 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos1_h' =	cSpos_h

Table 20: Event Table for cSpeicher_Pos1_hor

Name	Mode Class
cSpeicher_Pos1_hor	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_1 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos1_hor' =	cSpos_hor

Table 21: Event Table for cSpeicher_Pos1_s

Name	Mode Class
cSpeicher_Pos1_s	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_1 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos1_s' =	cSpos_s

Table 22: Event Table for cSpeicher_Pos1_v

Name	Mode Class
cSpeicher_Pos1_v	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_1 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos1_v' =	cSpos_v

Table 23: Event Table for cSpeicher_Pos1_w

Name	Mode Class
cSpeicher_Pos1_w	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_1 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos1_w' =	cSpos_w

Table 24: Event Table for cSpeicher_Pos2_h

Name	Mode Class
cSpeicher_Pos2_h	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_2 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos2_h' =	cSpos_h

Table 25: Event Table for cSpeicher_Pos2_hor

Name	Mode Class
cSpeicher_Pos2_hor	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_2 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos2_hor' =	cSpos_hor

Table 26: Event Table for cSpeicher_Pos2_s

Name	Mode Class
cSpeicher_Pos2_s	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_2 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos2_s' =	cSpos_s

Table 27: Event Table for cSpeicher_Pos2_v

Name	Mode Class
cSpeicher_Pos2_v	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_2 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos2_v' =	cSpos_v

Table 28: Event Table for cSpeicher_Pos2_w

Name	Mode Class
cSpeicher_Pos2_w	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_2 = TRUE) AND (mMgmt_set = TRUE)
cSpeicher_Pos2_w' =	cSpos_w

Table 29: Event Table for cSpos_h

Name		Mode Class
cSpos_h		Sitzeinstellung
Modes	**Events**	
Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(Taktgeber = Elapsed) When ((cSmot_h = heben) AND (mBlockiert_h = FALSE))	@T(Taktgeber = Elapsed) When ((cSmot_h = senken) AND (mBlockiert_h = FALSE))
cSpos_h' =	cSpos_h + MOVEMENT	cSpos_h - MOVEMENT

Table 30: Event Table for cSpos_hor

Name		Mode Class
cSpos_hor		Sitzeinstellung
Modes	**Events**	
Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(Taktgeber = Elapsed) When ((cSmot_hor = nach_hinten) AND (mBlockiert_hor = FALSE))	@T(Taktgeber = Elapsed) When ((cSmot_hor = nach_vorne) AND (mBlockiert_hor = FALSE))
cSpos_hor' =	cSpos_hor + MOVEMENT	cSpos_hor - MOVEMENT

Table 31: Event Table for cSpos_s

Name		Mode Class
cSpos_s		Sitzeinstellung
Modes	**Events**	
Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(Taktgeber = Elapsed) When ((cSmot_s = weiter) AND (mBlockiert_s = FALSE))	@T(Taktgeber = Elapsed) When ((cSmot_s = enger) AND (mBlockiert_s = FALSE))
cSpos_s' =	cSpos_s + MOVEMENT	cSpos_s - MOVEMENT

Table 32: Event Table for cSpos_v

Name		Mode Class
cSpos_v		Sitzeinstellung
Modes	**Events**	
Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(Taktgeber = Elapsed) When ((cSmot_v = heben) AND (mBlockiert_v = FALSE))	@T(Taktgeber = Elapsed) When ((cSmot_v = senken) AND (mBlockiert_v = FALSE))
cSpos_v' =	cSpos_v + MOVEMENT	cSpos_v - MOVEMENT

Table 33: Event Table for cSpos_w

Name		Mode Class
cSpos_w		Sitzeinstellung
Modes	**Events**	
Sitz_bewegen_1, Sitz_bewegen_2, BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	@T(Taktgeber = Elapsed) When ((cSmot_w = flacher) AND (mBlockiert_w = FALSE))	@T(Taktgeber = Elapsed) When ((cSmot_w = steiler) AND (mBlockiert_w = FALSE))
cSpos_w' =	cSpos_w + MOVEMENT	cSpos_w - MOVEMENT

Table 34: Event Table for cWerte_gespeichert_1

Name	Mode Class
cWerte_gespeichert_1	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_1 = TRUE) AND (mMgmt_set = TRUE)
cWerte_gespeichert_1' =	TRUE

Table 35: Event Table for cWerte_gespeichert_2

Name	Mode Class
cWerte_gespeichert_2	Sitzeinstellung
Modes	**Events**
Tuer_offen, Ruhen	@T(mMgmt_2 = TRUE) AND (mMgmt_set = TRUE)
cWerte_gespeichert_2' =	TRUE

Table 36: Event Table for cZu_Beifahrertsg_M_pos_mgmt_1

Name		Mode Class
cZu_Beifahrertsg_M_pos_mgmt_1		Sitzeinstellung
Modes	**Events**	
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	@T(mMgmt_1 = TRUE) AND (cWerte_gespeichert_1 = TRUE)	@T(mMgmt_1 = FALSE)
cZu_Beifahrertsg_M_pos_mgmt_1' =	TRUE	FALSE

Table 37: Event Table for cZu_Beifahrertsg_M_pos_mgmt_2

Name		Mode Class
cZu_Beifahrertsg_M_pos_mgmt_2		Sitzeinstellung
Modes	**Events**	
Ruhen, Sitz_bewegen_1, Sitz_bewegen_2, Tuer_offen, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken, BM_Funksender	@T(mMgmt_2 = TRUE) AND (cWerte_gespeichert_2 = TRUE)	@T(mMgmt_2 = FALSE)
cZu_Beifahrertsg_M_pos_mgmt_2' =	TRUE	FALSE

Table 38: Event Table for cZu_Beifahrertsg_M_save_1

Name		Mode Class
cZu_Beifahrertsg_M_save_1		Sitzeinstellung
Modes	**Events**	
Tuer_offen, Ruhen	@T(mMgmt_1 = TRUE) AND (mMgmt_set = TRUE)	@T(mMgmt_1 = FALSE)
cZu_Beifahrertsg_M_save_1' =	TRUE	FALSE

Table 39: Event Table for cZu_Beifahrertsg_M_save_2

Name		Mode Class
cZu_Beifahrertsg_M_save_2		Sitzeinstellung
Modes	**Events**	
Tuer_offen, Ruhen	@T(mMgmt_2 = TRUE) AND (mMgmt_set = TRUE)	@T(mMgmt_2 = FALSE)
cZu_Beifahrertsg_M_save_2' =	TRUE	FALSE

Table 40: Event Table for tAktuelle_Speicher_Position (Sheet 1 of 2)

Name		Mode Class	
tAktuelle_Speicher_Position		Sitzeinstellung	
Modes	**Events**		
Tuer_offen, Ruhen, Sitz_bewegen_1, Sitz_bewegen_2	(@T(cIn_M_pos_1=TRUE) WHEN ((tSpeicher_Position1_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT))) OR (@T(mMgmt_1 = TRUE) AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT))	(@T(cIn_M_pos_2=TRUE) WHEN ((tSpeicher_Position2_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT))) OR (@T(mMgmt_2 = TRUE) AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT))	@T(mMgmt_1 = FALSE) OR @T(mMgmt_2 = FALSE)
BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	(@T(cIn_M_pos_1=TRUE) WHEN ((tSpeicher_Position1_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT))) OR (@T(mMgmt_1 = TRUE) AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT))	(@T(cIn_M_pos_2=TRUE) WHEN ((tSpeicher_Position2_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT))) OR (@T(mMgmt_2 = TRUE) AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT))	@T(tSpeicher_Position1_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 1) OR @T(tSpeicher_Position2_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 2) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v > V_SCHWELLWERT)

Table 40: Event Table for tAktuelle_Speicher_Position (Sheet 2 of 2)

Name		Mode Class	
tAktuelle_Speicher_Position		Sitzeinstellung	
Modes	**Events**		
BM_Funksender	(@T(cIn_M_pos_1=TRUE) WHEN ((tSpeicher_Position1_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT))) OR (@T(mMgmt_1 = TRUE) AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT))	(@T(cIn_M_pos_2=TRUE) WHEN ((tSpeicher_Position2_abgleich = FALSE) AND (mIn_Fzg_v=0) AND (mBatt >= BATTERIE_SCHWELLWERT))) OR (@T(mMgmt_2 = TRUE) AND (mIn_Fzg_v <= V_SCHWELLWERT) AND (mBatt >= BATTERIE_SCHWELLWERT))	@T(tSpeicher_Position1_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 1) OR @T(tSpeicher_Position2_abgleich = TRUE) WHEN (tAktuelle_Speicher_Position = 2) OR @T(mBatt < BATTERIE_SCHWELLWERT) OR @T(mIn_Fzg_v >0)
tAktuelle_Speicher_Position' =	1	2	0

Table 41: Event Table for tBM_Blockierung_aufgetreten_h

Name		Mode Class
tBM_Blockierung_aufgetreten_h		Sitzeinstellung
Modes	**Events**	
BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	((@T(Taktgeber = Elapsed) WHEN (cSmot_h!=ruhen)) AND (NOT @C(cSpos_h) WHEN (cSmot_h!=ruhen)))	((@T(Taktgeber = Elapsed) WHEN (cSmot_h!=ruhen)) AND (@C(cSpos_h) WHEN (cSmot_h!=ruhen)))
Ruhen, Tuer_offen	NEVER	@T(Taktgeber = Elapsed) AND (cSmot_h=ruhen)
tBM_Blockierung_aufgetreten_h' =	TRUE	FALSE

Table 42: Event Table for tBM_Blockierung_aufgetreten_hor

Name		Mode Class
tBM_Blockierung_aufgetreten_hor		Sitzeinstellung
Modes	**Events**	
BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	((@T(Taktgeber = Elapsed) WHEN (cSmot_hor!=ruhen)) AND (NOT @C(cSpos_hor) WHEN (cSmot_hor!=ruhen)))	((@T(Taktgeber = Elapsed) WHEN (cSmot_hor!=ruhen)) AND (@C(cSpos_hor) WHEN (cSmot_hor!=ruhen)))
Ruhen, Tuer_offen	NEVER	@T(Taktgeber = Elapsed) AND (cSmot_hor=ruhen)
tBM_Blockierung_aufgetreten_hor' =	TRUE	FALSE

Table 43: Event Table for tBM_Blockierung_aufgetreten_s

Name		Mode Class
tBM_Blockierung_aufgetreten_s		Sitzeinstellung
Modes	**Events**	
BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	((@T(Taktgeber = Elapsed) WHEN (cSmot_s!=ruhen)) AND (NOT @C(cSpos_s) WHEN (cSmot_s!=ruhen)))	((@T(Taktgeber = Elapsed) WHEN (cSmot_s!=ruhen)) AND (@C(cSpos_s) WHEN (cSmot_s!=ruhen)))
Ruhen, Tuer_offen	NEVER	@T(Taktgeber = Elapsed) AND (cSmot_s=ruhen)
tBM_Blockierung_aufgetreten_s' =	TRUE	FALSE

Table 44: Event Table for tBM_Blockierung_aufgetreten_v

Name		Mode Class
tBM_Blockierung_aufgetreten_v		Sitzeinstellung
Modes	**Events**	
BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	((@T(Taktgeber = Elapsed) WHEN (cSmot_v!=ruhen)) AND (NOT @C(cSpos_v) WHEN (cSmot_v!=ruhen)))	((@T(Taktgeber = Elapsed) WHEN (cSmot_v!=ruhen)) AND (@C(cSpos_v) WHEN (cSmot_v!=ruhen)))
Ruhen, Tuer_offen	NEVER	@T(Taktgeber = Elapsed) AND (cSmot_v=ruhen)
tBM_Blockierung_aufgetreten_v' =	TRUE	FALSE

Table 45: Event Table for tBM_Blockierung_aufgetreten_w

Name		Mode Class
tBM_Blockierung_aufgetreten_w		Sitzeinstellung
Modes	**Events**	
BM_Funksender, BM_Sitz_bewegen_entspannen, BM_Sitz_bewegen_einschraenken	((@T(Taktgeber = Elapsed) WHEN (cSmot_w!=ruhen)) AND (NOT @C(cSpos_w) WHEN (cSmot_w!=ruhen)))	((@T(Taktgeber = Elapsed) WHEN (cSmot_w!=ruhen)) AND (@C(cSpos_w) WHEN (cSmot_w!=ruhen)))
Ruhen, Tuer_offen	NEVER	@T(Taktgeber = Elapsed) AND (cSmot_w=ruhen)
tBM_Blockierung_aufgetreten_w' =	TRUE	FALSE

Table 46: Condition Table for tBM_einschraenken_h

Name		
tBM_einschraenken_h		
	Conditions	
	((tBM_einschraenken_v = FALSE AND tBM_einschraenken_s = FALSE AND tBM_einschraenken_w = FALSE) OR (tBM_einschraenken_v = FALSE AND tBM_einschraenken_s = FALSE AND tBM_einschraenken_hor = FALSE) OR (tBM_einschraenken_v = FALSE AND tBM_einschraenken_w = FALSE AND tBM_einschraenken_hor = FALSE) OR (tBM_einschraenken_s = FALSE AND tBM_einschraenken_w = FALSE AND tBM_einschraenken_hor = FALSE)) AND (tBM_Blockierung_aufgetreten_h = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND ((cSpos_h < cSpeicher_Pos1_h) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_h < cSpeicher_Pos2_h) AND (tAktuelle_Speicher_Position = 2)) AND (mIn_Fzg_v <= V_SCHWELLWERT)	NOT(((tBM_einschraenken_v = FALSE AND tBM_einschraenken_s = FALSE AND tBM_einschraenken_w = FALSE) OR (tBM_einschraenken_v = FALSE AND tBM_einschraenken_s = FALSE AND tBM_einschraenken_hor = FALSE) OR (tBM_einschraenken_v = FALSE AND tBM_einschraenken_w = FALSE AND tBM_einschraenken_hor = FALSE) OR (tBM_einschraenken_s = FALSE AND tBM_einschraenken_w = FALSE AND tBM_einschraenken_hor = FALSE)) AND (tBM_Blockierung_aufgetreten_h = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND ((cSpos_h < cSpeicher_Pos1_h) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_h < cSpeicher_Pos2_h) AND (tAktuelle_Speicher_Position = 2)) AND (mIn_Fzg_v <= V_SCHWELLWERT))
tBM_einschraenken_h =	TRUE	FALSE

Table 47: Condition Table for tBM_einschraenken_hor

Name		
tBM_einschraenken_hor		
	Conditions	
	((cSpos_hor > cSpeicher_Pos1_hor) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_hor > cSpeicher_Pos2_hor) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_Blockierung_aufgetreten_hor = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND (mIn_Fzg_v <= V_SCHWELLWERT)	NOT(((cSpos_hor > cSpeicher_Pos1_hor) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_hor > cSpeicher_Pos2_hor) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_Blockierung_aufgetreten_hor = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND (mIn_Fzg_v <= V_SCHWELLWERT))
tBM_einschraenken_hor =	TRUE	FALSE

Table 48: Condition Table for tBM_einschraenken_s

Name		
tBM_einschraenken_s		
	Conditions	
	(tBM_einschraenken_hor = FALSE OR tBM_einschraenken_w = FALSE) AND (tBM_Blockierung_aufgetreten_s = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND ((cSpos_s > cSpeicher_Pos1_s) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_s > cSpeicher_Pos2_s) AND (tAktuelle_Speicher_Position = 2)) AND (mIn_Fzg_v <= V_SCHWELLWERT)	NOT((tBM_einschraenken_hor = FALSE OR tBM_einschraenken_w = FALSE) AND (tBM_Blockierung_aufgetreten_s = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND ((cSpos_s > cSpeicher_Pos1_s) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_s > cSpeicher_Pos2_s) AND (tAktuelle_Speicher_Position = 2)) AND (mIn_Fzg_v <= V_SCHWELLWERT))
tBM_einschraenken_s =	TRUE	FALSE

Table 49: Condition Table for tBM_einschraenken_v

Name		
tBM_einschraenken_v		
	Conditions	
	((tBM_einschraenken_hor = FALSE AND tBM_einschraenken_s = FALSE) OR (tBM_einschraenken_hor = FALSE AND tBM_einschraenken_w = FALSE) OR (tBM_einschraenken_w = FALSE AND tBM_einschraenken_s = FALSE)) AND (tBM_Blockierung_aufgetreten_v = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND (((cSpos_v < cSpeicher_Pos1_v) AND (tAktuelle_Speicher_Position = 1)) OR ((cSpos_v < cSpeicher_Pos2_v) AND (tAktuelle_Speicher_Position = 2))) AND (mln_Fzg_v <= V_SCHWELLWERT)	NOT(((tBM_einschraenken_hor = FALSE AND tBM_einschraenken_s = FALSE) OR (tBM_einschraenken_hor = FALSE AND tBM_einschraenken_w = FALSE) OR (tBM_einschraenken_w = FALSE AND tBM_einschraenken_s = FALSE)) AND (tBM_Blockierung_aufgetreten_v = FALSE) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND (((cSpos_v < cSpeicher_Pos1_v) AND (tAktuelle_Speicher_Position = 1)) OR ((cSpos_v < cSpeicher_Pos2_v) AND (tAktuelle_Speicher_Position = 2))) AND (mln_Fzg_v <= V_SCHWELLWERT))
tBM_einschraenken_v =	TRUE	FALSE

Table 50: Condition Table for tBM_einschraenken_w

Name		
tBM_einschraenken_w		
	Conditions	
	((cSpos_w > cSpeicher_Pos1_w) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_w > cSpeicher_Pos2_w) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND (tBM_Blockierung_aufgetreten_w = FALSE) AND (mln_Fzg_v <= V_SCHWELLWERT)	NOT(((cSpos_w > cSpeicher_Pos1_w) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_w > cSpeicher_Pos2_w) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_h = FALSE AND tBM_entspannen_hor = FALSE) AND (tBM_Blockierung_aufgetreten_w = FALSE) AND (mln_Fzg_v <= V_SCHWELLWERT))
tBM_einschraenken_w =	TRUE	FALSE

Table 51: Condition Table for tBM_entspannen_h

Name		
tBM_entspannen_h		
	Conditions	
	((tBM_entspannen_v = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_w = FALSE) OR (tBM_entspannen_v = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_hor = FALSE) OR (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_hor = FALSE) OR (tBM_entspannen_s = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_hor = FALSE)) AND (tBM_Blockierung_aufgetreten_h = FALSE) AND ((cSpos_h > cSpeicher_Pos1_h) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_h > cSpeicher_Pos2_h) AND (tAktuelle_Speicher_Position = 2)) AND (mIn_Fzg_v <= V_SCHWELLWERT)	NOT(((tBM_entspannen_v = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_w = FALSE) OR (tBM_entspannen_v = FALSE AND tBM_entspannen_s = FALSE AND tBM_entspannen_hor = FALSE) OR (tBM_entspannen_v = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_hor = FALSE) OR (tBM_entspannen_s = FALSE AND tBM_entspannen_w = FALSE AND tBM_entspannen_hor = FALSE)) AND (tBM_Blockierung_aufgetreten_h = FALSE) AND ((cSpos_h > cSpeicher_Pos1_h) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_h > cSpeicher_Pos2_h) AND (tAktuelle_Speicher_Position = 2)) AND (mIn_Fzg_v <= V_SCHWELLWERT))
tBM_entspannen_h =	TRUE	FALSE

Table 52: Condition Table for tBM_entspannen_hor

Name		
tBM_entspannen_hor		
	Conditions	
	((cSpos_hor < cSpeicher_Pos1_hor) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_hor < cSpeicher_Pos2_hor) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_Blockierung_aufgetreten_hor = FALSE) AND (mln_Fzg_v <= V_SCHWELLWERT)	NOT(((cSpos_hor < cSpeicher_Pos1_hor) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_hor < cSpeicher_Pos2_hor) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_Blockierung_aufgetreten_hor = FALSE) AND (mln_Fzg_v <= V_SCHWELLWERT))
tBM_entspannen_hor =	TRUE	FALSE

Table 53: Condition Table for tBM_entspannen_s

Name		
tBM_entspannen_s		
	Conditions	
	(tBM_entspannen_hor = FALSE OR tBM_entspannen_w = FALSE) AND (tBM_Blockierung_aufgetreten_s = FALSE) AND ((cSpos_s < cSpeicher_Pos1_s) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_s < cSpeicher_Pos2_s) AND (tAktuelle_Speicher_Position = 2)) AND (mln_Fzg_v <= V_SCHWELLWERT)	NOT((tBM_entspannen_hor = FALSE OR tBM_entspannen_w = FALSE) AND (tBM_Blockierung_aufgetreten_s = FALSE) AND ((cSpos_s < cSpeicher_Pos1_s) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_s < cSpeicher_Pos2_s) AND (tAktuelle_Speicher_Position = 2)) AND (mln_Fzg_v <= V_SCHWELLWERT))
tBM_entspannen_s =	TRUE	FALSE

Table 54: Condition Table for tBM_entspannen_v

Name		
tBM_entspannen_v		
	Conditions	
	((tBM_entspannen_hor = FALSE AND tBM_entspannen_s = FALSE) OR (tBM_entspannen_hor = FALSE AND tBM_entspannen_w = FALSE) OR (tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE)) AND (tBM_Blockierung_aufgetreten_v = FALSE) AND (((cSpos_v > cSpeicher_Pos1_v) AND (tAktuelle_Speicher_Position = 1)) OR ((cSpos_v > cSpeicher_Pos2_v) AND (tAktuelle_Speicher_Position = 2))) AND (mIn_Fzg_v <= V_SCHWELLWERT)	NOT(((tBM_entspannen_hor = FALSE AND tBM_entspannen_s = FALSE) OR (tBM_entspannen_hor = FALSE AND tBM_entspannen_w = FALSE) OR (tBM_entspannen_w = FALSE AND tBM_entspannen_s = FALSE)) AND (tBM_Blockierung_aufgetreten_v = FALSE) AND (((cSpos_v > cSpeicher_Pos1_v) AND (tAktuelle_Speicher_Position = 1)) OR ((cSpos_v > cSpeicher_Pos2_v) AND (tAktuelle_Speicher_Position = 2))) AND (mIn_Fzg_v <= V_SCHWELLWERT))
tBM_entspannen_v =	TRUE	FALSE

Table 55: Condition Table for tBM_entspannen_w

Name		
tBM_entspannen_w		
	Conditions	
	((cSpos_w < cSpeicher_Pos1_w) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_w < cSpeicher_Pos2_w) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_Blockierung_aufgetreten_w = FALSE) AND (mIn_Fzg_v <= V_SCHWELLWERT)	NOT(((cSpos_w < cSpeicher_Pos1_w) AND (tAktuelle_Speicher_Position = 1) OR (cSpos_w < cSpeicher_Pos2_w) AND (tAktuelle_Speicher_Position = 2)) AND (tBM_Blockierung_aufgetreten_w = FALSE) AND (mIn_Fzg_v <= V_SCHWELLWERT))
tBM_entspannen_w =	TRUE	FALSE

Table 56: Event Table for tHinten_bewegen

Name		
tHinten_bewegen		
	Events	
	(@T(mSitz_h = heben) WHEN (cSpos_h != Maximalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE)) OR (@T(mSitz_h = senken) WHEN (cSpos_h != Minimalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE))	@T(mSitz_h = ruhen) WHEN (cSmot_h!=ruhen) OR @T(cSpos_h = Maximalwert) OR @T(cSpos_h = Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_h!=ruhen)) AND (NOT @C(cSpos_h) WHEN (cSmot_h!=ruhen))) OR @T(mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT)
tHinten_bewegen' =	TRUE	FALSE

Table 57: Event Table for tHor_bewegen

Name		
tHor_bewegen		
	Events	
	(@T(mSitz_hor = nach_hinten) WHEN (cSpos_hor != Maximalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE)) OR (@T(mSitz_hor = nach_vorne) WHEN (cSpos_hor != Minimalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE))	@T(mSitz_hor = ruhen) WHEN (cSmot_hor != ruhen) OR @T(cSpos_hor = Maximalwert) OR @T(cSpos_hor = Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_hor!=ruhen)) AND (NOT @C(cSpos_hor) WHEN (cSmot_hor!=ruhen))) OR @T(mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT)
tHor_bewegen' =	TRUE	FALSE

Table 58: Event Table for tSchalung_bewegen

Name		
tSchalung_bewegen		
	Events	
	(@T(mSitz_s = weiter) WHEN (cSpos_s != Maximalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE)) OR (@T(mSitz_s = enger) WHEN (cSpos_s != Minimalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE))	@T(mSitz_s = ruhen) WHEN (cSmot_s != ruhen)OR @T(cSpos_s = Maximalwert) OR @T(cSpos_s = Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_s!=ruhen)) AND (NOT @C(cSpos_s) WHEN (cSmot_s!=ruhen))) OR @T(mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT)
tSchalung_bewegen' =	TRUE	FALSE

Table 59: Condition Table for tSitzeinstellung_aktiviert

Name		
tSitzeinstellung_aktiviert		
	Conditions	
	(CONFIG_SITZ_F = TRUE AND ((IN_LL = TRUE))) OR ((CONFIG_SITZ_BF = TRUE) AND ((IN_RL = TRUE)))	NOT ((CONFIG_SITZ_F = TRUE AND IN_LL = TRUE) OR (CONFIG_SITZ_BF = TRUE AND IN_RL = TRUE))
tSitzeinstellung_aktiviert =	TRUE	FALSE

Table 60: Condition Table for tSpeicher_Position1_abgleich

Name		
tSpeicher_Position1_abgleich		
	Conditions	
	(cSpos_h=cSpeicher_Pos1_h OR tBM_Blockierung_aufgetreten_h = TRUE) AND (cSpos_hor=cSpeicher_Pos1_hor OR tBM_Blockierung_aufgetreten_hor = TRUE) AND (cSpos_w=cSpeicher_Pos1_w OR tBM_Blockierung_aufgetreten_w = TRUE) AND (cSpos_v=cSpeicher_Pos1_v OR tBM_Blockierung_aufgetreten_v = TRUE) AND (cSpos_s=cSpeicher_Pos1_s OR tBM_Blockierung_aufgetreten_s = TRUE)	NOT((cSpos_h=cSpeicher_Pos1_h OR tBM_Blockierung_aufgetreten_h = TRUE) AND (cSpos_hor=cSpeicher_Pos1_hor OR tBM_Blockierung_aufgetreten_hor = TRUE) AND (cSpos_w=cSpeicher_Pos1_w OR tBM_Blockierung_aufgetreten_w = TRUE) AND (cSpos_v=cSpeicher_Pos1_v OR tBM_Blockierung_aufgetreten_v = TRUE) AND (cSpos_s=cSpeicher_Pos1_s OR tBM_Blockierung_aufgetreten_s = TRUE))
tSpeicher_Position1_abgleich =	TRUE	FALSE

Table 61: Condition Table for tSpeicher_Position2_abgleich

Name		
tSpeicher_Position2_abgleich		
	Conditions	
	(cSpos_h=cSpeicher_Pos2_h OR tBM_Blockierung_aufgetreten_h = TRUE) AND (cSpos_hor=cSpeicher_Pos2_hor OR tBM_Blockierung_aufgetreten_hor = TRUE) AND (cSpos_w=cSpeicher_Pos2_w OR tBM_Blockierung_aufgetreten_w = TRUE) AND (cSpos_v=cSpeicher_Pos2_v OR tBM_Blockierung_aufgetreten_v = TRUE) AND (cSpos_s=cSpeicher_Pos2_s OR tBM_Blockierung_aufgetreten_s = TRUE)	NOT((cSpos_h=cSpeicher_Pos2_h OR tBM_Blockierung_aufgetreten_h = TRUE) AND (cSpos_hor=cSpeicher_Pos2_hor OR tBM_Blockierung_aufgetreten_hor = TRUE) AND (cSpos_w=cSpeicher_Pos2_w OR tBM_Blockierung_aufgetreten_w = TRUE) AND (cSpos_v=cSpeicher_Pos2_v OR tBM_Blockierung_aufgetreten_v = TRUE) AND (cSpos_s=cSpeicher_Pos2_s OR tBM_Blockierung_aufgetreten_s = TRUE))
tSpeicher_Position2_abgleich =	TRUE	FALSE

Table 62: Event Table for tVorne_bewegen

Name		
tVorne_bewegen		
	Events	
	(@T(mSitz_v = heben) WHEN (cSpos_v != Maximalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE)) OR (@T(mSitz_v = senken) WHEN (cSpos_v != Minimalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE))	@T(mSitz_v = ruhen) WHEN (cSmot_v != ruhen) OR @T(cSpos_v = Maximalwert) OR @T(cSpos_v = Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_v!=ruhen)) AND (NOT @C(cSpos_v) WHEN (cSmot_v!=ruhen))) OR @T(mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT)
tVorne_bewegen' =	TRUE	FALSE

Table 63: Event Table for tWinkel_bewegen

Name		
tWinkel_bewegen		
	Events	
	(@T(mSitz_w = flacher) WHEN (cSpos_w != Maximalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE)) OR (@T(mSitz_w = steiler) WHEN (cSpos_w != Minimalwert AND mBatt >= BATTERIE_SCHWELLWERT AND mT_offen = TRUE))	@T(mSitz_w = ruhen) WHEN (cSmot_w != ruhen) OR @T(cSpos_w = Maximalwert) OR @T(cSpos_w = Minimalwert) OR ((@T(Taktgeber = Elapsed) WHEN (cSmot_w!=ruhen)) AND (NOT @C(cSpos_w) WHEN (cSmot_w!=ruhen))) OR @T(mT_offen = FALSE) OR @T(mBatt < BATTERIE_SCHWELLWERT)
tWinkel_bewegen' =	TRUE	FALSE

Literaturverzeichnis

[TSG] “Beispiel-Systemspezifikation: Türsteuergerät”, Version 1.0, Frank Houdek, 5.2.2001

Dokumenten Information

Titel:	Modellierung eines Türsteuergerätes mit SCR
Datum:	22. Oktober 2001
Report:	IESE-064.01/D
Status:	Final
Verteiler:	Public