Bewegungssteuerungen auf Basis des Hybriden Funktionsplanes[1)]

1. Einleitung

Während in der Elektrotechnik zur Beschreibung von Steuerungsabläufen und Bewegungsvorgängen meist Zeitablaufdiagramme eingesetzt werden, stellt auf dem Gebiet des Maschinenbaus das Arbeitsdiagramm in der Regel die Ausgangsbasis zur Konstruktion einer neuen Maschine dar. Läßt man auf der x-Achse als Bezugsgröße sowohl die Zeit, als auch den Drehwinkel zu, so können die beiden Arbeitsmittel als äquivalent aufgefaßt werden. Bei konstanter Drehzahl einer Welle sind Zeit- und Winkelfunktion vergleichbar.

Der Aufsatz skizziert in gestraffter Form die Vorgehensweise bei der Entwicklung einer Bewegungssteuerung als ereignisdiskretes, zeitkontinuierliches System ausgehend vom Zeitablaufdiagramm über die graphische Programmierung und Dokumentation mit Hybriden Funktionsplänen mittels der SIMULINK Erweiterungs-Blockbibliothek „Funktionsplan“ (FUP) einschließlich Test und Simulation bei Verwendung von Maschinenmodellen bis hin zur Generierung von Maschinenkode in typisch 7 Schritten. Der Einsatz von FUP unter SIMULINK bietet dem Anwender Vorteile wie:

- Richtliniennähe
- Graphische Programmierung im SIMULINK *Standardfenster*
- Anwendung der *bekannten* SIMULINK *Bedienung*
- Simulation *ohne Kompilierung*

Wesentliche Details des Entwurfsablaufes werden durch ausgewählte Beispiele veranschaulicht.

2. Ausgangspunkt Arbeitsdiagramm

Bei Verarbeitungsmaschinen mit zentralem Antrieb [Sta94] werden die Bewegungen der einzelnen Arbeitsorgane von der Drehbewegung der zentralen Programmwelle abgeleitet. Das ist möglich, weil zwischen der Abtriebsbewegung des antreibenden Motors und den mit Mechanismen erzeugten Einzelbewegungen der Arbeitsorgane festkörpermechanischer Zwanglauf besteht. Die feste Zuordnung der Bewegungen ist in *allen* Betriebszuständen gegeben, also außer im Nennbetrieb, auch beim An- und Abfahren, im Tippbetrieb, bei Notaus oder bei Verstellungen der Betriebsdrehzahl und damit der Arbeitstaktzahl. Die Darstellung der Bewegungsdiagramme erfolgt zweckmäßigerweise nicht als Funktion der Zeit, sondern als Funktion des Drehwinkels der Programmwelle. Stellt man diese Bewegungsdiagramme zusammen und ergänzt sie durch die während der einzelnen Bewegungsabschnitte verrichteten Arbeitsvorgänge sowie durch die den Arbeitsablauf bestimmenden Ereignisse, so erhält man das *Arbeitsdiagramm.* Bei Verstellungen, z.B. bei Formatänderung, ändern sich die Bewegungsdiagramme, so daß Kurvenscharen entstehen. Diese Arbeitsdiagramme beschreiben anschaulich graphisch das *Arbeitsprogramm* einer

[1)] Erweiterte und geänderte Fassung des Aufsatzes: Geitner, G.-H.; Stange, H. „Die steuerungstechnische Umsetzung des Arbeitsdiagramms mit dem Hybriden Funktionsplan“ zur VVD 2000, S. 214-233, Proceedings ISBN 3-86005-247-0

Maschine und haben deshalb zentrale Bedeutung für die Funktionsbeschreibung einer Verarbeitungsmaschine, sowohl bei Entwurf und Konstruktion einer neuen Maschine, als auch bei der Maschinenanalyse für die Bewertung, Verbesserung, Weiterentwicklung oder Fehlersuche. Beim Entwurf stellt das Arbeitsdiagramm die durch die Konstruktion zu realisierende Sollfunktion dar, möglichst ergänzt durch einzuhaltende Toleranzschranken. Für die Analyse wird das Arbeitsdiagramm häufig quasistatisch aufgenommen, wodurch dynamische Einflüsse nicht berücksichtigt werden können. Besser ist die meßtechnische Erfassung, die jedoch meist als Zeitfunktion erfolgt. Nur bei konstanter Drehzahl der Programmwelle ist Zeit- und Winkelfunktion vergleichbar. Oft sorgen jedoch winkelabhängige Massenträgheits- und Lastmomente für Drehzahlschwankungen, die durch den Ungleichmäßigkeitsgrad ausgedrückt werden können.

Für eine intermittierend arbeitende Schlauchbeutel-Form-, Füll- und Verschließmaschine mit zentralem Antrieb sind Arbeitsverfahren und Arbeitsdiagramm beispielhaft in Bild 1 und Bild 2 angegeben. Dieses Beispiel wurde gewählt, weil derartige Verpackungsmaschinen weit verbreitet sind, die Anzahl der Arbeitsorgane und damit die Komplexität der Maschine gering ist und die Flexibilitätsanforderungen bezüglich Formatverstellung einen Übergang vom zentralen zum dezentralen Antriebskonzept nahe legen [Wei95].

Im Arbeitsdiagramm Bild 2 der zentral angetriebenen Verpackungsmaschine sind außer den Bewegungen der Arbeitsorgane auch Ereignisse und Vorgänge eingetragen, die zusammengenommen das Arbeitsprogramm der Maschine ausmachen. Diese Eintragungen haben bei *Zentralantrieben* lediglich erläuternde, das Verständnis erhellende Bedeutung. Durch den festkörpermechanischen Zwanglauf und damit verbundene eindeutige Zuordnungen der Lage der Arbeitsorgane untereinander, d.h. Weg- und Winkelstellung, werden keine steuerungstechnischen Befehle wie Ereignissignale, Zustandserfassungen und Ablaufstrukturen benötigt.

Bei Verarbeitungsmaschinen mit *dezentralen Antrieben* [Sch96] wird jedem Arbeitsorgan ein separater Antriebsstrang, der bezüglich Wandlung und Umformung von Energie möglichst gut an die kinetischen und energetischen Arbeitsorgananforderungen angepaßt ist, zugeordnet. Eine zentrale Programmwelle erübrigt sich. Es gibt keinen mechanischen Zwanglauf mehr. Die erforderliche Zuordnung und Koordination von Arbeitsorganbewegungen für aufgabengemäßes Zusammenspiel und Kollisionsvermeidung muß durch eine Bewegungssteuerung gewährleistet werden.

Die funktionellen Abläufe sind hier grundsätzlich zeitabhängig darzustellen, so daß mit dem Verlassen des Zwanglaufs auch die sehr bequeme und übersichtliche Winkelabhängigkeit bei der Darstellung des Arbeitsdiagramms verlassen werden muß. Dabei kann das Arbeitsdiagramm zunächst in einem ersten Schritt vereinfacht, qualitativ als Zeitfunktion dargestellt werden. Parameteränderungen lassen sich durch Änderungen der Zeitfunktionen berücksichtigen. In einigen Fällen kann auch die Darstellung über einer normierten Zeit hilfreich sein. Bezugsgröße ist in diesen Fällen die Taktzeit, also die Zeit für einen Arbeitstakt.

Entscheidende Bedeutung für das Zusammenspiel der Arbeitsorgane und damit für den "elektronischen Zwanglauf" ist jedoch, daß aus dem Arbeitsdiagramm zeitdiskrete Signale abgeleitet werden, die jeweils bestimmte Ereignisse in der Arbeits- und Antriebsabfolge der Maschine melden und Zustandswechsel veranlassen. Darstellungsmittel für die Abfolge der einzelnen Zustände und folglich für die Ablaufsteuerung ist der Funktionsplan (DIN 40719-6), der jedoch prinzipiell nur binäre Zustände kennt.

3. Hybrider Funktionsplan

3.1 Begriffsbestimmung

Funktionspläne beschreiben eine sequentielle Abfolge von Zuständen und Ereignissen. Eine Syntax zu alternativen und parallelen Verzweigungen und Zusammenführungen erlaubt dabei die Definition von sowohl sich ausschließenden, als auch gleichzeitig aktiven Zuständen. Die Zustände werden als konstant vorausgesetzt. Projektierbar sind Bewegungsabläufe, die eine Folge diskreter Zustände darstellen bzw. einen zeitproportionalen Übergang zwischen Grenzzuständen einschließen. Der Hybride Funktionsplan [VDI97], [Sch98] ist eine Verallgemeinerung insofern, als ein Zustand als beliebige (vektorielle) Zeitfunktion aufgefaßt wird. Bewegungsabläufe, auch von großem Umfang, sind folglich als zeitkontinuierlich-ereignisdiskrete Prozesse beschreibbar. Hierzu werden den Zuständen Aktionen zugeordnet, die auch die Realisierung nichtlinearer Zeitfunktionen einschließen. Aus den Zustandsgrößen des Prozesses sind *die* Ereignisse zu generieren, die ein Weiterschalten im Hybriden Funktionsplan bewirken, wobei gegebenenfalls eine Verknüpfung über Boolesche Gleichungen möglich ist. Falls erforderlich, können auch Ausgangssignale von Zeitgebern einbezogen werden.
Die Realisierung von Programmierwerkzeugen nach VDI/VDE Richtlinie 3684 [VDI97] schließt jedoch ebenso wenig wie die Norm IEC 61131-3 [Bau01] Mehrdeutigkeiten aus. Das betrifft z.B. den Programmfortschritt, die Abarbeitungsreihenfolge bei Verzweigungen oder Auswertungsprioritäten bei hierarchischen Systemen.

3.2 Anwendungsbeispiel

Die Beschreibung mehrdimensionaler zeitkontinuierlich-ereignisdiskreter Bewegungssteuerungen mit dezentralen Antrieben durch Hybride Funktionspläne kann mit der Symbolbibliothek nach Bild 3 erfolgen. Tabelle 1 faßt kurze Erläuterungen der durch die Blöcke realisierten Funktionen zusammen. Als Beispiel zur Demonstration der Anwendung dieser Beschreibungsmethode dient die in Abschnitt 2 vorgestellte Schlauchbeutel-form-, Füll- und -Verschließmaschine. Bild 4a und Bild 4b geben zum Vergleich eine zweidimensionale Darstellung des Arbeitsprinzips sowie ein Arbeitsdiagramm mit simuliertem Anfahrvorgang und Übergang in den Normalbetrieb an. Weitere

Beispiele und Erläuterungen zur in Abschnitt 4 skizzierten praktischen Realisierung unter SIMULINK sind unter [Web 1], [Web 9], [Web 10] und [Web 11] verfügbar.

Ein Hybrider Funktionsplan zur Bewegungssteuerung dieser vertikalen Schlauchbeutelmaschine ist vereinfacht in Bild 5a enthalten. Er setzt sich aus den vier sequentiellen Teilfunktionsplänen „Referenzpunktfahrt“, „Anlauf“, „Normalbetrieb“ und „Stillsetzen“ zusammen. Die dezentralen Antriebe [Gei99] für Bobine (Abwickler), Abzug u. Längssiegeln, Quersiegeln und Dosierung werden durch die Sollwerte v_1 bis v_4 gesteuert. Die Beeinflussung dieser Sollwerte geschieht graphisch verbindungslos durch die in den Teilfunktionsplänen jeweils aktiven Zustände – vgl. Bild 5b Normalbetrieb. Wobei die Sollwertvorgaben der den Zustandsblöcken zugeordneten Kommentarblöcke bei Zustandswechsel aktiviert werden. Für Antrieb 3 der Quersiegelbacken ist zur Untersuchung des Anfahrverhaltens zusätzlich ein zufälliger Anfangswinkel in Grad einstellbar. Vereinfachungen bestehen u.a. in der ausschließlichen Steuerung der Antriebe über Drehzahlsollwerte. Das ist jedoch im Hinblick auf die Darstellung der grundsätzlichen Vorgehensweise zulässig.

Die Ausgangsgrößen der Antriebe, Drehzahlistwerte „vist_x“ der Antriebe 1 bis 3, Lageistwerte „xist_y“ der Antriebe 3 und 4 sowie zusätzlich ein Nullimpuls „NI_3“ von Antrieb 3, bilden den (Prozeß-)Zustandsvektor „ZV“. Der Steuervektor „SV“ setzt sich aus den Ausgängen der Schalter „Referenzpunktsuche“ und „Hochlauf“ zur Auswahl unterschiedlicher Anfahrvarianten zusammen. Die Elemente der Eingangssignale „ZV“ und „SV“ des Blocks Ereignisgenerierung werden zur Erzeugung von Weiterschaltereignissen für den Hybriden Funktionsplan ausgewertet. Die Übertragung der in Teilereignisvektoren „EV1“ bis „EV4“ zusammengefaßten Ereignisse zu den Teilfunktionsplänen erfolgt graphisch verbindungslos – vgl. Bild 5b. Block AZF mit Parametrierung „Aktive Zustände FUP 1“ steuert den Standardblock „Display“ zur Anzeige der jeweils aktiven Zustände des Bewegungsablaufes an.

Mit Schalter „S3“ können Fehler während des Betriebs der Maschine simuliert werden. Das Signal „SAMMELFEHLER“ führt zur Aktivierung von Block ZF "Allgemeiner Fehler" und in der Folge zur sofortigen Deaktivierung aller aktiven Zustände des Normalablaufes. Dementsprechend können die (Teil-) Funktionspläne für den normalen Ablauf und für die Beseitigung von Fehlern völlig getrennt voneinander und ohne Beeinträchtigung der graphischen Übersichtlichkeit entworfen und programmiert werden. Das ist möglich, weil der Übergang zur Fehlerbehandlung graphisch verbindungslos erfolgt. Beispielhaft werden mit Block ZF „Allgemeiner Fehler“ (Zustand X200) alle vier Antriebssteuergrößen Null gesetzt. Ist der Teilfunktionsplan „Fehlerbeseitigung“ durchlaufen, wird „SAMMELFEHLER“ mit Schalter „S4“ quittiert und die Maschine nimmt den normalen Ablauf wieder auf.

Als Reaktion auf einen Fehler können je nach Vorzustand unterschiedliche Abläufe sinnvoll sein. Hierzu wird unmittelbar nach Block ZF eine alternative Verzweigung vorgesehen und das Weiterschalten, d.h. die Auswahl eines Pfades, von den Ergebnissen je einer in einem Block FVZ hinterlegten Booleschen Gleichung abhängig gemacht – Bild 5c. Die Elemente dieser logischen Glei-

chungen entsprechen Zustandsnummern des normalen Ablaufes und sind wahr, wenn der entsprechende Zustand vor dem Auftreten des Fehlers aktiv war. Nachfolgend sind prinzipiell weitere Verzweigungen wieder in Abhängigkeit von Prozeßzustandsgrößen möglich. Im Beispiel wird das Durchlaufen von einem der Fehlerkorrekturpfade für Anfahren, Normalbetrieb oder Stillsetzen mit „Quit1“, „Quit2“ oder „Quit3“ abgeschlossen und der Teilfunktionsplan Fehlerbeseitigung über den Zustand X111 „Endkontrolle“ verlassen. Da die Erläuterung des Prinzips im Vordergrund steht, sind den Zuständen X200, X300, X400 und X111 keine Sollwerte v_1 bis v_4 zugeordnet.

3.3 Vom Arbeitsdiagramm zum Hybriden Funktionsplan

Die vereinfachte Realisierung des stationären Betriebszustands der unter Abschnitt 2 beschriebenen Schlauchbeutelmaschine ist als Teilfunktionsplan „Normalbetrieb“ in Bild 5b dargestellt. Vereinfachend wird u.a. angenommen, daß der Dosiervorgang stets länger dauert als das Siegeln der Quernaht und ein Zustand X6 "Erster Quernahtsiegel- und Dosiervorgang" gefordert wird. Davon ausgehend führt Ereignis ES_9 "Siegelende" zum Eintritt in die Schleife des Normalbetriebs. Diese Schleife besteht im Beispiel aus der zyklischen Abfolge von Zustand X7 „Dosieren u. Siegelbacken beschleunigen", Ereignis ES_8 „Dosierende", Zustand X8 "Siegelbacken abbremsen und Dosierstop", Ereignis ES_10 "Siegelbeginn" (Verknüpfung von Winkel „x31“ mit negiertem Stopsignal), Zustand X70 "Quernahtsiegeln u. Dosieren" und Ereignis ES_9 "Siegelende". Ein Verlassen ist nach Abbremsen der Quersiegelbacken über ES_11 (Verknüpfung von Winkel „x31“ mit aktivem STOP Signal) vorgesehen. Die einzelnen Ereignisse werden dem Teilereignisvektor „EV3“ entnommen und über Konnektoren zum Angriffspunkt des Ereignisses übertragen. Die den Zuständen zugeordneten und in Blöcken KB definierten Aktionen werden mit Aktivierung des jeweiligen Zustandes ausgelöst und bleiben bis zur nächsten Veränderung gültig. Das Beispiel demonstriert Vorgehensweise und Möglichkeiten mit dem Werkzeug Hybrider Funktionsplan. Eine weitere Detaillierung erfordert auch genauere Prozeßmodelle.

4. Steuerungstechnische Umsetzung

4.1 Auswahl einer Basissoftware

Für die praktische Anwendung des Hybriden Funktionsplanes sind Softwarewerkzeuge zur graphischen Programmierung der Bewegungssteuerung, zur Suche von Parameter- und Syntaxfehlern und zur automatischen Generierung von Maschinenkode notwendig. Prinzipiell kann hierzu eine eigenständige Software entwickelt werden. Dem Vorteil dieses Lösungsweges, der maximalen Anpassung an die Aufgabenstellung, stehen jedoch grundsätzliche Nachteile wie aufwendige Entwicklung und Pflege eines x-ten graphischen Editors oder fehlende Möglichkeiten zur Prozeßmodellierung gegenüber. Wesentlich vorteilhafter ist die Auswahl und Erweiterung einer eingeführten, weit verbreiteten und offenen Softwarelösung mit Werkzeugen zur Simulation und Ma-

schinenkodegenerierung. Das damit die Möglichkeiten der graphischen Gestaltung fest vorgegeben sind, muß keinen schwerwiegenden Nachteil darstellen.

Das ingenieurtechnische Programmpaket MATLAB / SIMULINK [Web 2] stellt u. a. auf Grund folgender Eigenschaften eine gute Basis dar:

- Erweiterungsmöglichkeit der offenen, blockorientierten Simulationsoberfläche SIMULINK zur Modellierung kontinuierlicher, diskontinuierlicher und hybrider Prozesse mit *gleichberechtigten* Nutzer- / Erweiterungs-Blockbibliotheken.
- Komfortabler graphischer Editor mit drag & drop Technologie, Aktionsspeicher und strukturierter offengelegter Modellbeschreibung.
- Programmierung von Anwenderfunktionalitäten in C, FORTRAN oder der MATLAB eigenen Sprache und Aufruf als Funktion in SIMULINK-Anwenderblöcken oder eigenständig am MATLAB Prompt.
- Umfangreiche Möglichkeiten zur funktionellen und graphischen Gestaltung von Anwenderblöcken wie Blockmaskierung, Parameterdialog, Anfangswertberechnung, Hilfetexte ... .
- Praktisch unbegrenzte Definition von horizontalen und vertikalen Teilsystemen, sowie bedingte Ausführung von Teilsystemen, d. h. Freigabe- und Triggersteuerung.
- Zahlreiche Werkzeuge zur Unterstützung von Programmentwicklung und Fehlersuche wie eingebauter Editor, Debugger sowohl für MATLAB, als auch für SIMULINK, eingebaute, einfache Generierung von „dynamic link libraries“ (dll, mexw32, mexw64, mexhpux, …) – vgl. [Scw08] S. 503, Toolboxen und Blöcke zur Auswertung von Signalen, Steuerung des Gültigkeitsbereiches von Variablen, Festkommabibliothek, manuelle online Schalter ... .

Auf Grund der aufgelisteten, sowie zahlreicher weiterer Vorteile und der Verfügbarkeit sowohl unter Windows, als auch unter UNIX, LINUX und Macintosh wurde MATLAB / SIMULINK als Basis ausgewählt.

4.2 Maschinenkodeerzeugung mit der Basissoftware

Zur Erzeugung von Maschinenkode wird die Erweiterung Real-Time Workshop (RTW) benötigt. Mit RTW ist eine in SIMULINK graphisch programmierte Struktur, also z. B. eine Bewegungssteuerung, entworfen mittels Blöcken einer Erweiterungs-Blockbibliothek Hybrider Funktionsplan (FUP) unter Verwendung von SIMULINK Standard-Blocken, in Standard-ANSI-C Kode übersetzbar. Dieser Kode kann für jede Zielplattform, die eine Floating-Point-Unit hat und für die ein solcher Standard-ANSI-C-Kompiler existiert, lauffähig gemacht werden. Die Anbindung an die entsprechende Peripherie läßt sich leicht durch die Möglichkeit des Einbindens der spezifischen Treiberroutinen realisieren.

Durch ein weiteres Werkzeug, das CONTI-Tool, ist die Einschränkung auf rein integerbasierten Kode, einschließlich der Konvertierung von Parametern und Variablen, bei Bedarf möglich. Obwohl der Kodegenerierungsprozeß automatisiert abläuft, hat der Nutzer durch die Offenheit des Gesamtsystems in jedem Bereich der Kodeerzeugung die Möglichkeit einzugreifen und eigene Modifikationen vorzunehmen. Unter anderem sind für folgende Varianten vordefinierte, modifizierbare Softwarelösungen verfügbar:

- Kodegenerierung mit RTW und PCMON-Tool für PC-Einsteck-Karten (A/D, D/A) unter Windows am gleichen PC.
- Kodegenerierung mit RTW, CONTI-Tool und Cross-Kompiler für Mikrocontroller, z.B. Intel 80C196 oder Siemens C166/167.
- Kodegenerierung mit RTW und Fixed Point DSP Block Set für die DSP 56300 Familie.
- Kodegenerierung mit RTW und dSPACE-Software für Entwicklungssysteme mit TMS320C31, C30, C40 und Alpha 21164 Prozessor.
- Kodegenerierung mit RTW und xPC Target (vormals ReaILink/32) [Web 3] für PC kompatible Targetsysteme ohne DOS oder Windows.

Für die praktische Erprobung im Labor wurde die letzte Variante gewählt, da sie einerseits eine hohe Flexibilität und andererseits ein Optimum von Kosten und Abtastzeiten gewährleistet. Die Software xPC Target bietet eine umfangreiche Treiberbibliothek für Peripheriekarten von z. Z. *34 Herstellern* mit allen wesentlichen Funktionen wie: A/D-Wandlung, D/A-Wandlung; Digitaleingabe, Digitalausgabe; Generierung von PWM; Anschluß von Inkrementalgebern und Zählern, Schnittstellen für: serielle Kopplung, CAN-Bus, Ethernet, Speicher, Video, - vgl. [Web 3]. Ein Treiber für die PROFIBUS-Schnittstelle wurde mit Unterstützung der Fakultät Informatik am Institut entwickelt. Im Labor wurden die beiden Varianten „D/A-Wandler zur Vorgabe von Drehzahlsollwerten und Schnittstellen für Inkrementalgeber zur Messung von Drehzahlistwerten“ sowie „Kommunikation über PROFIBUS mit Nutzung antriebsinterner Messungen“ für drei Drehstrom-Masterdrive Antriebe erprobt – vgl. auch Bild 15 unten. Die studentische Ausbildung erfolgt mit letztgenannter Variante.

5. Projektierung mit FUP

5.1 Eigenschaften der SIMULINK Blockbibliothek FUP

Der in Abschnitt 3 vorgestellte Hybride Funktionsplan steht als Erweiterungs-Blockbibliothek FUP unter SIMULINK zur Verfügung [Gei01] und wird durch folgende Eigenschaften charakterisiert [Web 1]:

- Strikte Einhaltung der SIMULINK Philosophie (kein Spezialfenster mit abweichender Bedienung, keine Kompilierung bei Simulation, keine Kodierungssoftware).

- Konformität zur VDI/VDE Richtlinie 3684 [VDI97]. Geringfügige Abweichungen sind der Verwendung des SIMULINK Standard-Editors geschuldet.
- Graphisch übersichtliche Fehlerbehandlung durch separat programmierbare Fehlerbehandlungsabläufe ohne Einfluß auf die Übersichtlichkeit des Normalablaufes. In vielen Fällen wird der Wiedereintrittspunkt nach Fehlerbeseitigung dem Anfangszustand entsprechen. Andere technologisch sinnvolle Wiedereintrittszustände sind möglich – vgl. Bild 6.
- Keine Restriktionen hinsichtlich der Verschachtelung von parallelen und alternativen Verzweigungen und Zusammenführungen in einem Funktionsplan entsprechend der Syntax.
- Keine Restriktionen hinsichtlich der Bildung von Teilfunktionsplänen. Große Funktionspläne zur Programmierung von Bewegungssteuerungen können in bildschirm-kompatible Abschnitte aufgeteilt werden.
- Die Zustandsnummern sind frei wählbar. Es besteht *kein* Zusammenhang zwischen der SIMULINK internen, automatischen Blocknummerierung und den wählbaren Zustandsnummern.
- Durch Vorgabe von Funktionsplan- *und* Zustandsnummern besteht die Möglichkeit der gleichzeitigen graphischen Programmierung mehrerer Funktionspläne in einem Projekt bei Nutzung gleicher Zustandsnummern.

Für die aktuelle Version 3.1 von FUP gelten folgende Grenzwerte, die auch im zweisprachigen, entsprechend SIMULINK per Browser erreichbaren, online Hilfetext der betreffenden Bibliotheksblöcke nachlesbar sind:

- zulässige ganzzahlige Funktionsplannummern: $1 \leq FN \leq 9$
- zulässige ganzzahlige Zustandsnummern: $1 \leq BN \leq 9999$
- zulässige Länge für Namen- und Zahlenwertzeichenketten: 31
- zulässige Zeichenkettenlänge für Boolesche Gleichungen: 100

Diese Zahlenwerte stellen keine Grenzen für die steuerungstechnische Realisierung dar. Die Anzahl der mit *einem* Zustandswechsel definierbaren Aktionen (Steuersignale) ist grundsätzlich nicht limitiert.

5.2 Projektierungsschritte

Vor Bearbeitung des *ersten* konkreten Projektes müssen die Bibliotheksfiles von FUP (Version 3.1: drei Files *.lib) einmalig mit dem Zielhardwarekompiler erzeugt, in der Software zur Anpassung an die Zielhardware (z. B. xPC Target) angemeldet und gegebenenfalls gespeichert werden. Des Weiteren benötigt RTW einige spezielle Files (Version 3.1: vier Files) zur Steuerung der automatischen Kodegenerierung, Anpassung an den verwendeten Kompiler und Definition eines externen Interface. Im Fall der unter Abschnitt 4.2 aufgelisteten Softwarelösungen werden diese

Files mitgeliefert - anderenfalls muß eine Modifikation der entsprechenden RTW Standardfiles durch den Nutzer erfolgen.

Der Ablauf zur Generierung von Maschinenkode für eine konkrete Bewegungssteuerung läßt sich typisch in sieben Schritte unterteilen, die iterativ mehrmals durchlaufen werden. Der Anwender definiert zunächst eine Projektdirektorie mit wahlfreiem Namen, kopiert die C-Quellfiles (Version 3.1: drei *.C Files) sowie die Headerfiles (Version 3.1: sechs *.h Files) der FUP-Blockbibliothek in die Direktorie und stellt diese Direktorie als Arbeitsdirektorie ein. Danach wird nach den in Bild 7 angegebenen Schritten verfahren:

Schritt 1

Der geforderte technologische Ablauf wird entsprechend den Weiterschaltbedingungen im Arbeitsdiagramm graphisch mit Blöcken der FUP Blockbibliothek (Bild 3) durch Anwendung der drag & drop Technik programmiert. Zur Erhöhung von Übersichtlichkeit und Lesbarkeit kann von den hierarchischen Gestaltungsmöglichkeiten beliebig Gebrauch gemacht werden. Einfache Parametrierungsfehler kommen durch akustische Signale und Fehlertexte, wie unter SIMULINK üblich, sofort zur Anzeige. Im Ergebnis liegt eine vollständige Beschreibung der Bewegungssteuerung mittels Funktionsplan vor.

Schritt 2

Mit Zusatzsoftwarewerkzeugen [Web 4] kann im Projekt beginnend mit einer Analyse des FUP's (Zustandsanzahl etc.) automatisch nach falscher Parametrierung (Zustands- und Funktionsplannummern, Definition und Verwendung von Variablennamen, Zustandsnummern bei Abläufen zur Fehlerbehandlung), Syntaxfehlern (siehe Richtlinie), arithmetischen Schleifen (wesentlich bei Rückführungen, die nicht über den Anfangszustand führen) und Zustandssackgassen (erste / alle) gesucht werden. Bild 8 zeigt beispielhaft Prüfprotokolle bei Funktionsaufruf am MATLAB prompt. Wahlweise stehen hierfür jedoch GUI's (Bild 9) *oder* Skript-Files zur Verfügung. Ergebnisausgaben sind entweder in gestraffter Form oder ausführlich einschließlich von Hinweisen möglich. Die vielfältigen Prüfmöglichkeiten verdeutlicht Bild 10 anhand der Prüfschrittauswahl „Parameter". Bei Simulation der Ereignisse durch manuelle online Schalter können bei Bedarf ausführliche statische Tests erfolgen (Bild 11). Entwurfsschritt zwei liefert eine formal fehlerfreie, statisch ablauffähige Bewegungssteuerung.

Schritt 3

Umfangreiche dynamische Tests des Gesamtsystems (Bild 12a) mit Funktionsplan (Bild 12b) sind durch Erweiterung um Ereignisgenerierung (Bild 12c) und Prozeßmodell (Bild 12d) realisierbar. SIMULINK Standardbibliotheken stellen vielfältige Möglichkeiten zur graphischen Programmierung kontinuierlicher Regelstrecken und diskontinuierlicher Regelkreise zur Verfügung. Die Optimierung digitaler Regler ist z.B. auf Grundlage von [Gei96] mit der freien MATLAB Toolbox „BOD" [Web 5], [Web 6] möglich. Der Nutzer kann dabei zwischen der Definition von Übertragungsfunk-

tionen oder LTI-Modellen bzw. Menüführung oder Steuervektorvorgabe bei wahlweiser Abschaltung von Rückmeldungen wählen.
Das Verhalten interessierender Zustandsgrößen ist einfach dokumentierbar – vgl. Bild 4b. Gegebenenfalls kann eine Bedienoberfläche entworfen werden bzw. eine teilweise Kombination mit Schrittbetrieb über manuelle online Schalter erfolgen. Technologiefehler sind erkennbar, technologische Varianten gut vergleichbar. Im Ergebnis enthält die programmierte Bewegungssteuerung keine mittels Simulation durch Prozeßmodelle erkennbaren Fehler mehr. Die Genauigkeit des Prozeßmodells bestimmt jedoch wesentlich die Güte dieses Schrittes. Dieser Entwicklungsstand muß archiviert werden.

Schritt 4

Das Projektfile wird kopiert und die Ausgänge der Bewegungssteuerung (Prozeßschnittstellen) werden mit Blockeingängen der Treiber für Prozeßein / -ausgaben statt mit dem Streckenmodell verbunden (Bild 13). Das geschieht entweder durch Entfernen und Ersetzen des Modells durch Blöcke der Hardwaretreiberbibliothek oder durch eine Umschaltung zwischen diesen zwei Varianten. Die Treiber müssen parametriert werden - Abtastzeiten, Kanäle, Genauigkeit etc. Jetzt liegt eine vollständige Beschreibung der Bewegungssteuerung zur Umsetzung in Maschinenkode vor.

Schritt 5

Über ein Nutzerheaderfile (Bild 14) erfolgt die Optimierung des Speicherbedarfs der Echtzeitsteuerung, z. B. Einstellung der maximalen Anzahl paralleler Zustände in einem Funktionsplan, der maximalen Anzahl paralleler Funktionspläne in einem Projekt oder der maximalen Anzahl von Steuervariablen. Über Kompilerschalter können Kode und Abtastzeit durch Ausblendung der Erkennung von einfachen Fehlern nach Schritt 1 verkürzt werden. Die eingangs von Abschnitt 5.2 erwähnten Steuerfiles für RTW müssen über das SIMULINK Parametermenü eingetragen werden. Die Software-Lösungen bieten i. a. Werkzeuge hierfür (xPC Target: rl32initmodel). Damit sind die Vorarbeiten zur Generierung von Maschinenkode abgeschlossen.

Schritt 6

Nach Einstellung von Testparametern (Stopzeit, Solver etc.) wird die von einem Makefile gesteuerte Generierung des Echtzeitkodes durch eine Schaltfläche direkt im Bedienmenü des SIMULINK Projektfensters ausgelöst. Bei fehlerfreiem Lauf und eingeschalteter Zielhardware erfolgt über serielle Schnittstelle oder Ethernet das sofortige Laden des erzeugten Maschinenkodes in die Zielhardware (Target). Die Zielplattform ist damit testbereit (Bild 15). Der Datenaustausch mit den dezentralen Antrieben kann je nach Hardwarekonzept über ein Bussystem [Fra99] mit spezieller Steuerelektronik oder Definition eines Leitantriebes bzw. sternförmig über einen Steuerrechner mit E/A-Baugruppen erfolgen.

Schritt 7

Freigabe des Datenaustausches zwischen SIMULINK und Target. Mittels GUI's der Echtzeitkode Erstellungssoftware kann der Start des Maschinenkodes ausgelöst und der Test am realen Prozeß dokumentiert werden (xPC Target: rl32host, rl32scope). Online Schalter ermöglichen Variantenvergleiche. Einfache Möglichkeiten zur Beeinflussung des Testablaufes stehen über die GUI's und das SIMULINK Projektfenster zur Verfügung. Ein komfortables Bedienen und Beobachten ist über gegebenenfalls entworfene Bedienoberflächen (Schritt 3) möglich.

6. Ausblick

Die Umsetzung des Arbeitsdiagramms in einen Hybriden Funktionsplan wurde zunächst manuell vorgenommen. Eine weitgehende Automatisierung ist möglich [Blu00]. Mit zwischen Maschinenbau und Elektrotechnik fachübergreifenden Entwicklungswerkzeugen kann eine maschinelle Generierung von Hybriden Funktionsplänen aus dem Bewegungs-Design heraus erfolgen. Erste Versuche hierzu lassen interessante Ergebnisse erwarten. Der Entwurf von Verarbeitungsmaschinen auf Basis von dezentralen Einzelantrieben wird wesentlich beeinflusst werden. Eine weitere vielversprechende Anwendung kann in der Implementierung des verallgemeinerten OMAC Maschinen-Models bestehen [Web 7], [Web 8], [Hag08]. Bild 16 und Bild 17 geben eine Anregung hierzu.

Literatur

[Bau01] Bauer, N.; Treseler, H.: Vergleich der Semantik der Ablaufsprache nach IEC 61131-3 in unterschiedlichen Programmierwerkzeugen, VDI-Berichte Nr. 1608, 2001, S.135/42.

[Blu00] Blümel, R.: Entwurf dezentraler elektromechanischer Antriebe für Verarbeitungsmaschinen von den technologischen Anforderungen zum optimalen Antriebssystem, Dissertation, TU Dresden 2000.

[Fra99] Franke, M.: Intelligente Antriebe im Systemverbund. Ein Beitrag zur Untersuchung des Betriebsverhaltens von dezentralen feldbusgekoppelten Antrieben. VDI-Fortschritt- Berichte, R. 21, Nr. 227. Düsseldorf VDI-Verlag, 1999.

[Gei96] Geitner, G.-H.: Entwurf digitaler Regler für elektrische Antriebe, VDE Verlag, Berlin und Offenbach,1996.

[Gei99] Geitner G.-H.; Schönfeld R.: Distributed Motion Control - Use of Function Chart a SIMULINK 2.2 Add-on Block Library, 8th European Conference on Power Electronics and Applications - EPE '99. Lausanne, 07.-09. Sep. 1999, CD-ROM, 8 Seiten.

[Gei01] Geitner G.-H.: Die Anwendung der SIMULINK Blockbibliothek "Funktionsplan" V. 3.0 (FUP) zum Entwurf dezentraler Bewegungssteuerungen, 7. Fachtagung Engineering komplexer Automatisierungssysteme-EKA ´01, 25./27.04.01 Braunschweig, S.381/89.

[Hag08] Hagge, N.; Wagner, B.: Implementation Alternatives for the OMAC State Machines Using IEC 61499, Proc. IEEE Int. Conf. Emerging Technol. Factory Autom., 2008, S.215/220.

[Sch96] Schönfeld, R.; Stange, H.: Moderne Antriebssysteme in Verarbeitungsmaschinen. Tagungsband VVD 96, Verarbeitungsmaschinen und Verpackungstechnik Dresden 1996.

[Sch98] Schönfeld. R.; Geitner, G.-H.: Dezentrale Bewegungssteuerungen, Analyse und Entwurf mit dem Hybriden Funktionsplan, SPS/IPC/DRIVES, Nürnberg '98, Tagungsband, S. 335/43.

[Scw08] Schweizer, W.: MATLAB kompakt, Oldenbourg Verlag, 2008.

[Sta94] Stange, H.: Bewegungsanforderungen bei Verarbeitungsmaschinen, Tagungsband SPS / IPC / DRIVES '94, Sindelfingen, 1994.

[VDI97] VDI/VDE Richtlinie 3684: Beschreibung ereignisgesteuerter Bewegungsabläufe mit Funktionsplänen, Düsseldorf 1997.

[Web_1] Geitner, G.-H.: Entwurf, Simulation, Generierung von Echtzeitkode und Dokumentation ereignisgesteuerter Systeme mittels Hybrider Funktionspläne (FUP), http://eeiwzg.et.tu-dresden.de/ae2_files/ae_7.htm

[Web_2] MATLAB / SIMULINK - kommerzielle und nichtkommerzielle Links, http://eeiwzg.et.tu-dresden.de/ae2_files/ae_1_1.htm

[Web_3] xPC Target, http://www.mathworks.de/products/xpctarget

[Web_4] FupCheck - Toolbox Funktionsplanprüfung für FUP, http://eeiwzg.et.tu-dresden.de/ae2_files/ae_7_6.htm

[Web_5] BOD - Digitales Betragsoptimum, http://eeiwzg.et.tu-dresden.de/ae2_files/ae_1_0.htm

[Web_6] BOD Toolbox Version 2.5, http://eeiwzg.et.tu-dresden.de/ae2_files/ae_1_0_2.htm

[Web_7] http://www.omac.org/

[Web_8] http://www.packworld.com/controls/strategy/omac-standardizes-machine-states

[Web_9] Blockbibliothek FUP, http://eeiwzg.et.tu-dresden.de/ae2_files/ae_7_1.htm (sensitiv)

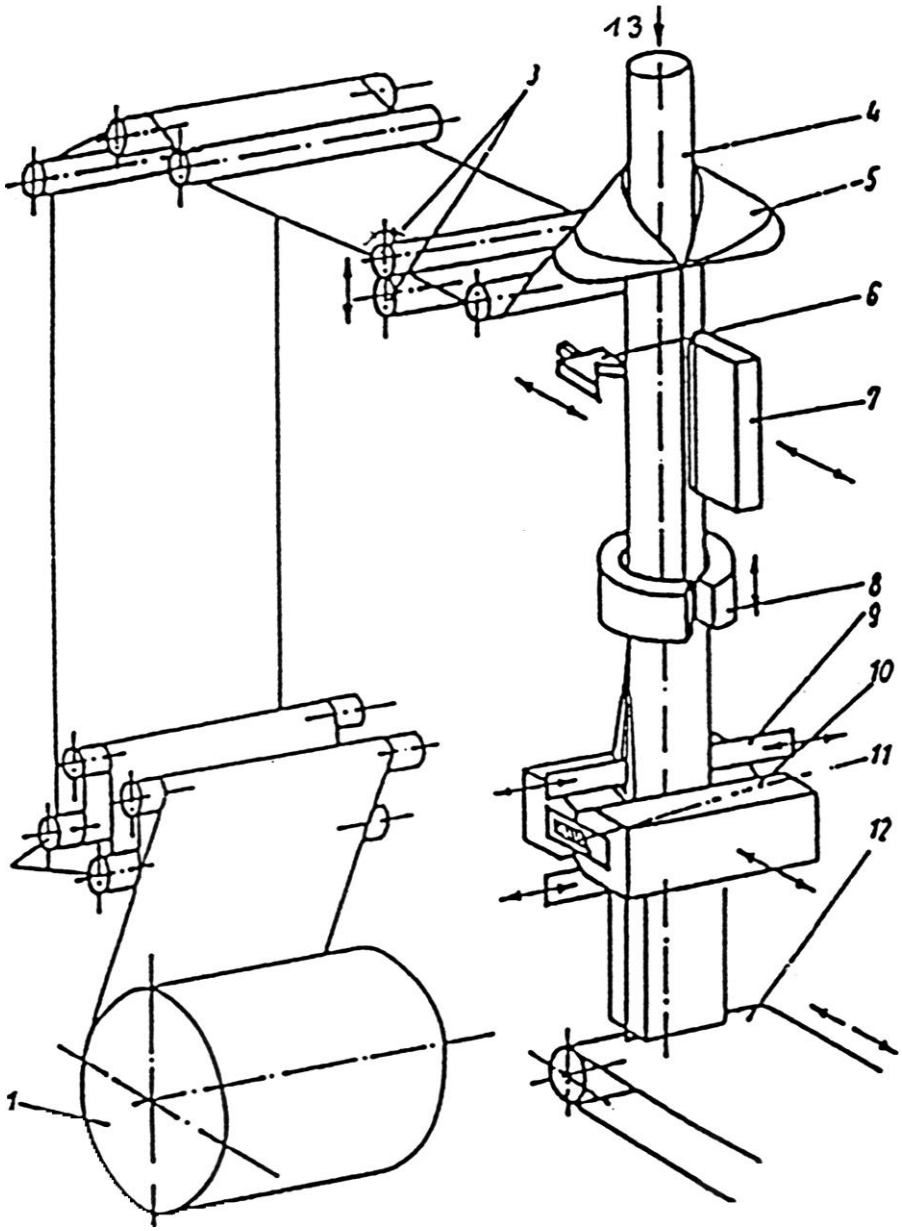
[Web_10] Blockbeschreibung FUP, http://eeiwzg.et.tu-dresden.de/ae2_files/ae_7_3.htm

[Web_11] Beispiele mit FUP, http://eeiwzg.et.tu-dresden.de/ae2_files/ae_7_5.htm (z. T. sensitiv)

[Wei95] Weile, F.: Komplexe Untersuchung von Antriebs- und Verarbeitungssystemen mittels Simulation am Beispiel von Schlauchbeutelmaschinen. Dissertation, TU Dresden 1995.

Kontakt

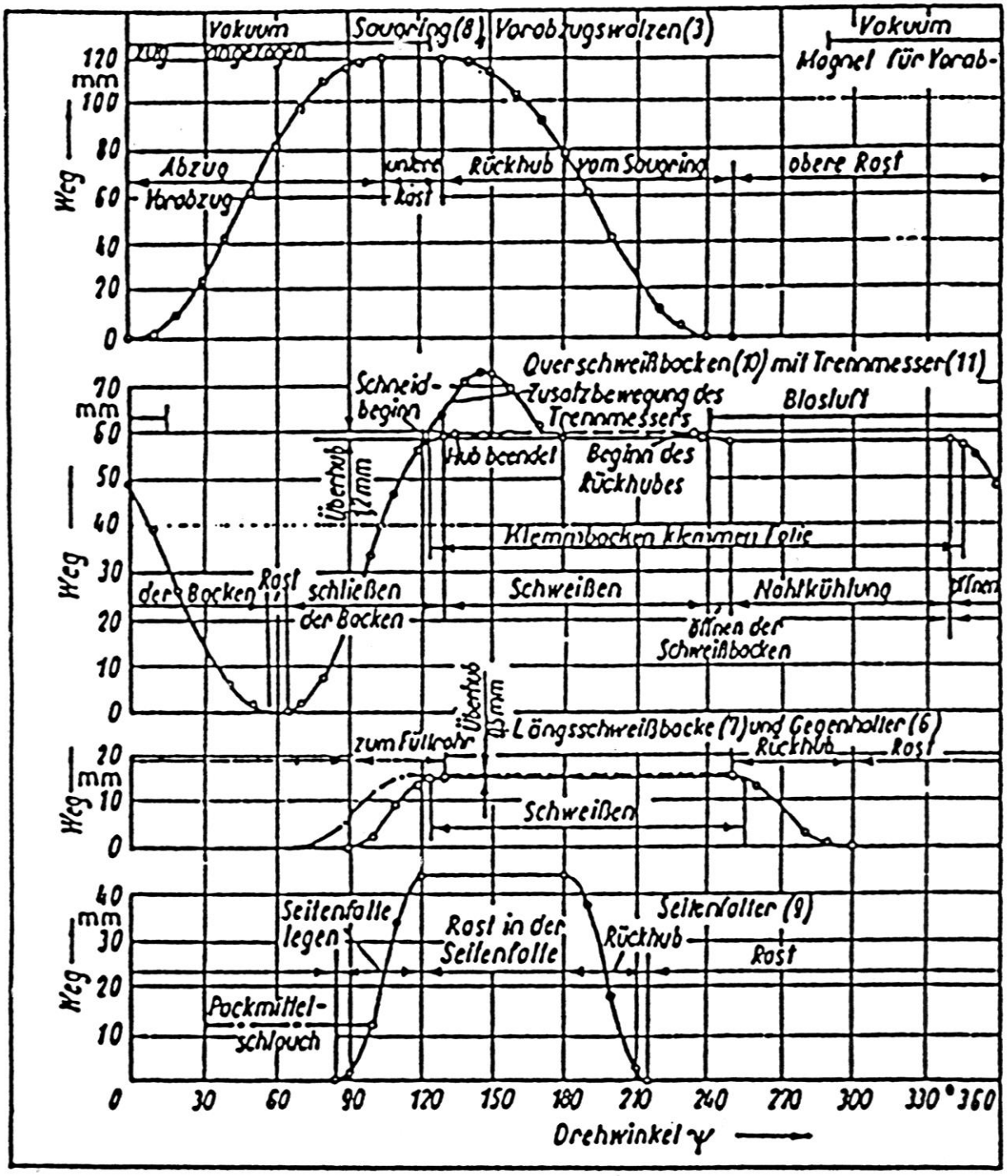
Priv.-Doz. Dr.-Ing. habil. Gert-Helge Geitner,
Technische Universität Dresden, Helmholtzstr. 10, 01062 Dresden; Fakultät EuI,
Elektrotechnisches Institut; Tel.: +49-351-463-32917, FAX: +49-351-463-33655,
e-mail: Gert-Helge.Geitner@tu-dresden.de; HP: http://eeiwzg.et.tu-dresden.de/ae2_files/ae_0.htm



Arbeitsverfahren einer Verpackungsmaschine
(Schlauchbeutel-Form-, Füll- und Verschließmaschine)

1 Packmittelrolle; 2 Abzugschwinge; 3 Vorabzugswalzen; 4 Formrohr;
5 Formschulter; 6 Gegenhalter;
7 dauerbeheizte Längssiegel- bzw. Längsschweißbacken;
8 Saugring für Packmittelabzug; 9 Seitenfalter;
10 dauerbeheizte Quersiegel- bzw. Querschweißbacken;
11 Trennmesser; 12 Transportband; 13 Füllgut

Bild 1 Beispiel für ein Arbeitsverfahren (Arbeitsprinzip) nach [Sta94], [Wei95]



Arbeitsdiagramm einer Verpackungsmaschine

(Schlauchbeutel-Form-, Füll- und Verschließmaschine)

Dargestellt sind die Bewegungen von Saugring, Quer- und Längsschweißbacken sowie Seitenfalter.

Bild 2 Beispiel für ein Arbeitsdiagramm (winkelabhängig) nach [Sta94], [Wei95]

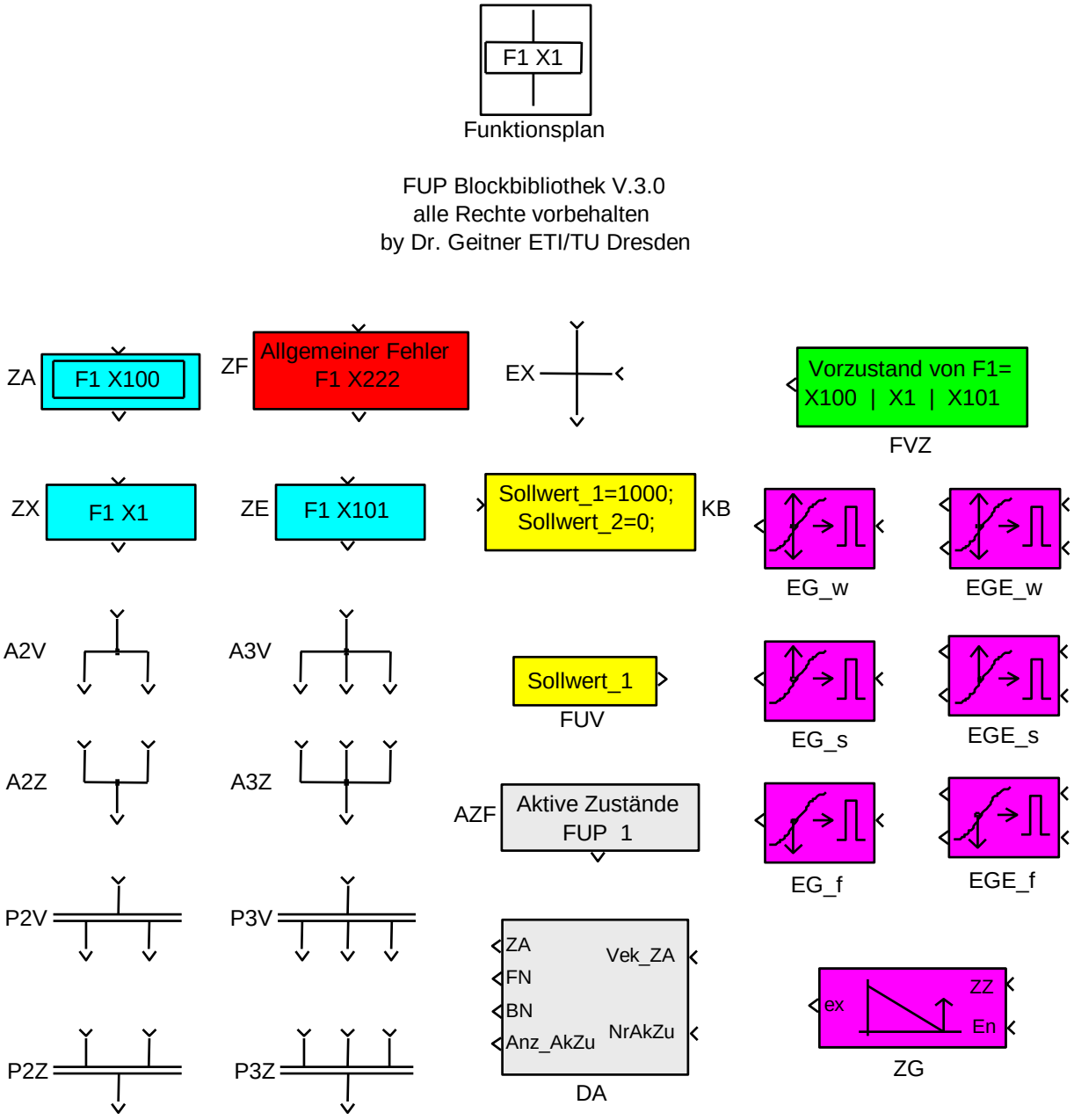
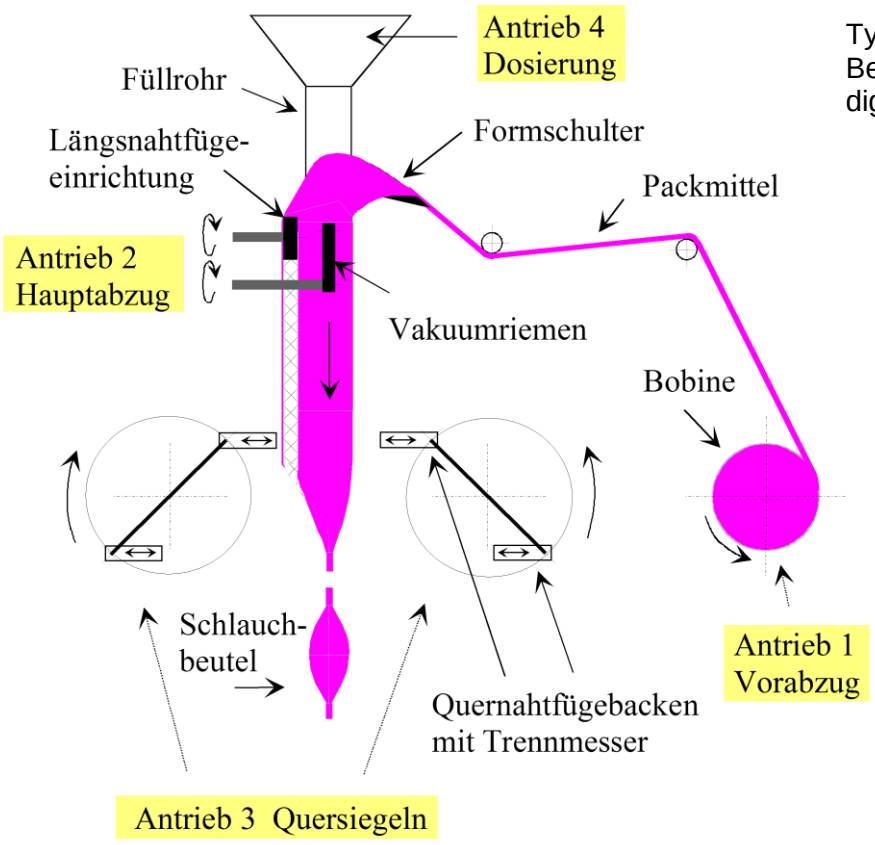


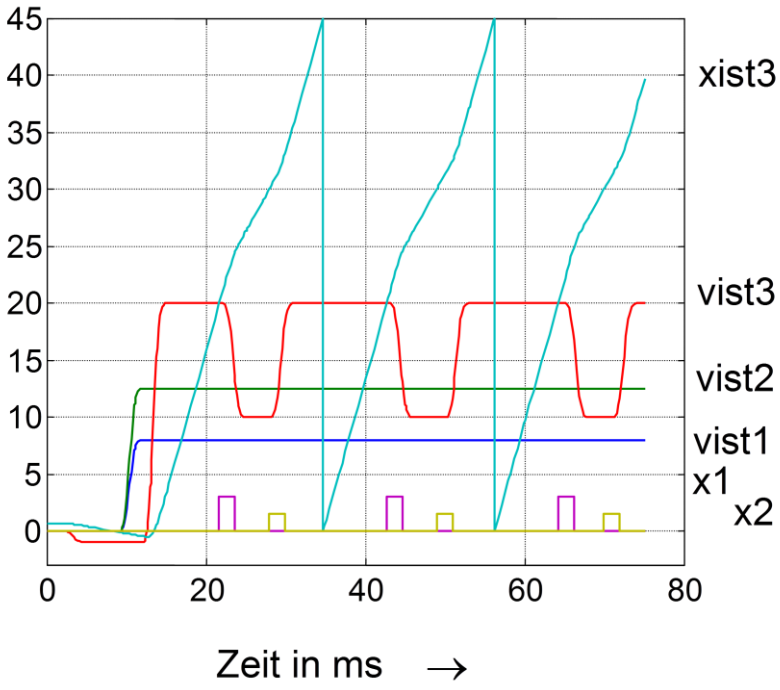
Bild 3 Simulink Erweiterungs-Blockbibliothek Hybrider Funktionsplan (FUP)
- Projektierungswerkzeug für ereignisdiskret-kontinuierliche Prozesse -
[Web_1], [Web_9] (sensitiv), [Web_10]



Typische mehrdimensionale Bewegung mit dezentralen digitalen Antrieben:

- Füllen
- Formen
- Abwickeln
- Verschließen

Bild 4a Arbeitsprinzip mit vier Antrieben - Beispiel Schlauchbeutelmaschine



Drehwinkel:
Quersiegeln

Drehzahlen:
Quersiegeln

Hauptabzug

Vorabzug

Ereignisse:
Siegelbeginn
und -ende

Bild 4b Projektierungswerkzeug Arbeitsdiagramm (Anfahren u. Normalbetrieb) - Beispiel Schlauchbeutelmaschine

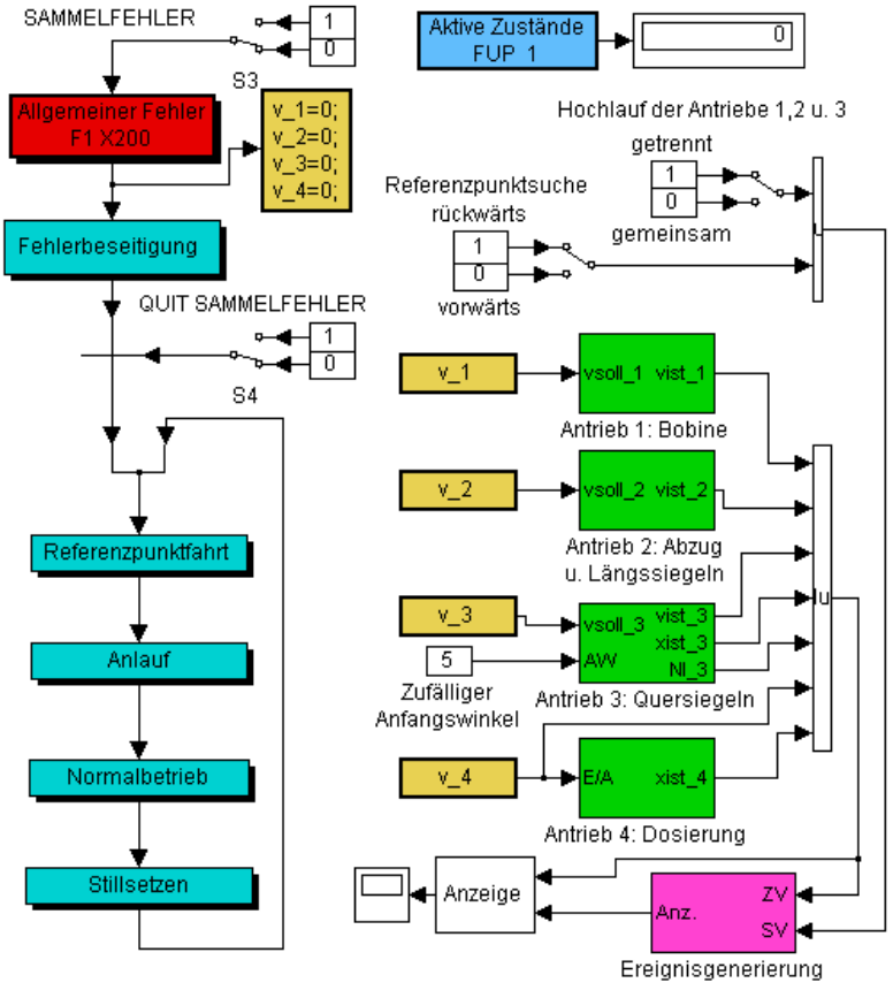


Bild 5a Aufteilung in Teilfunktionspläne

Beispiel: Simulation der Bewegungssteuerung einer Schlauchbeutelmaschine mit HFUP (vereinfacht), [Web_11-BEI9] (sensitiv)

ES_8=x33:
Winkel x33 - Antrieb 3 erreicht Dosierende

ES_9=x32:
Winkel x32 - Antrieb 3 erreicht Siegelende

ES_10=x31•/STOP bzw.
ES_11=x31•STOP:
Winkel x31 - Antrieb 3 erreicht Siegel- u. Dosierbeginn

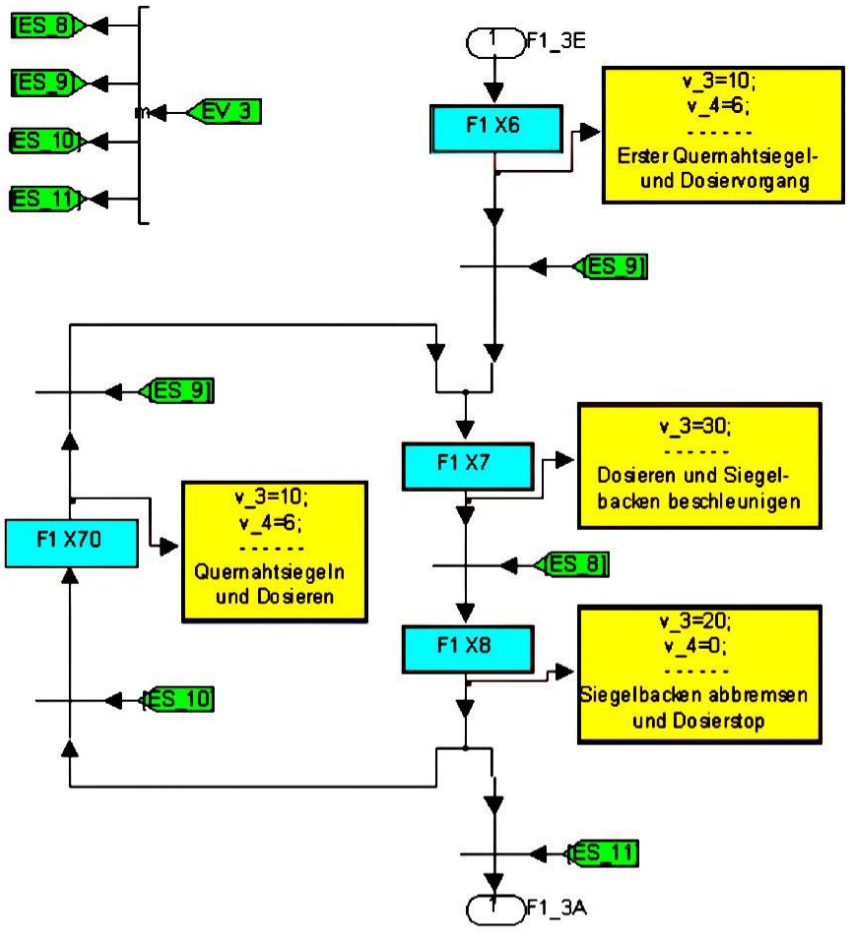


Bild 5b Definition von Teilfunktionsplänen

Beispiel: Teil-HFUP für den Normalbetrieb der Schlauchbeutelmaschine (vereinfacht), [Web_11-bei9_tf3]

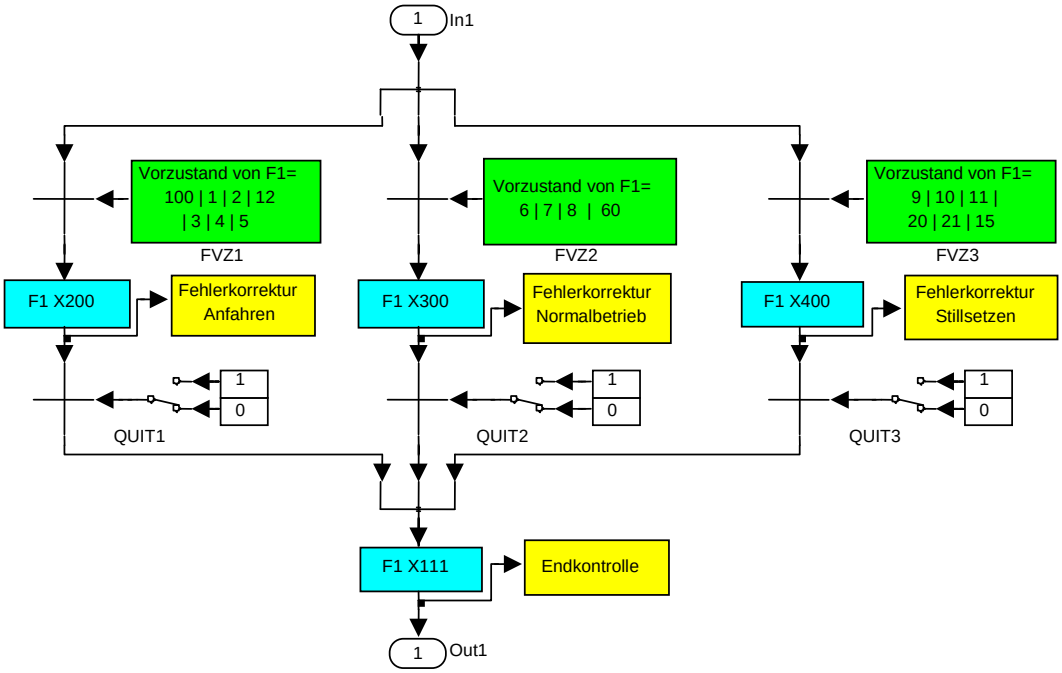


Bild 5c Verzweigte Fehlerbehandlung
Beispiel: Teilfunktionsplan Fehlerbeseitigung für Schlauchbeutelmaschine (vereinfacht)

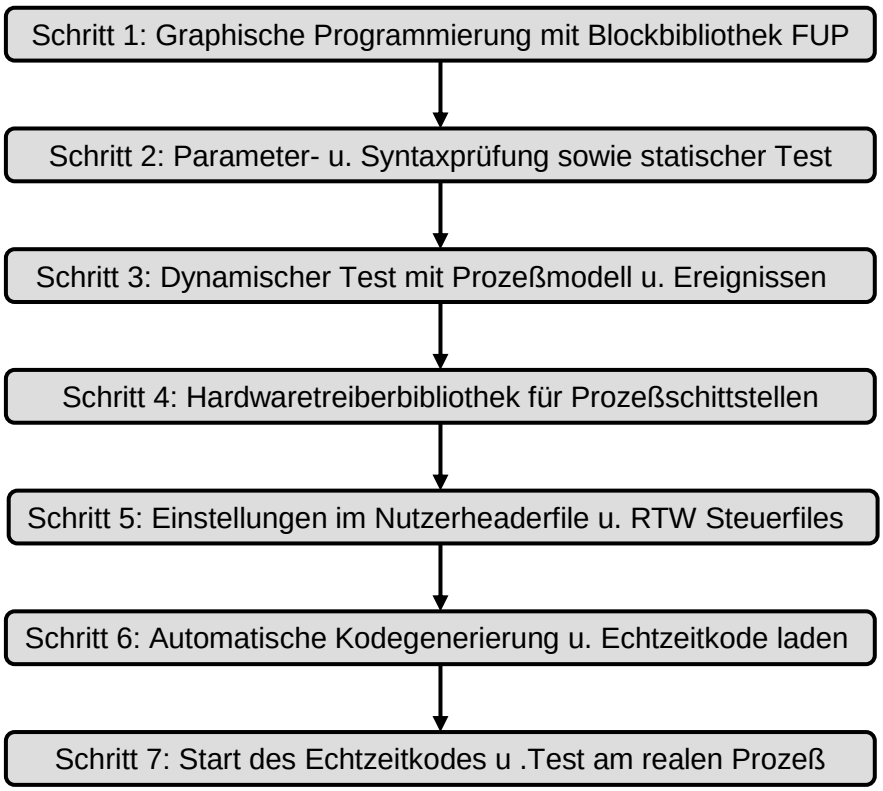


Bild 7 Entwurf in 7 Schritten - Vom Arbeitsdiagramm zum Maschinenkode

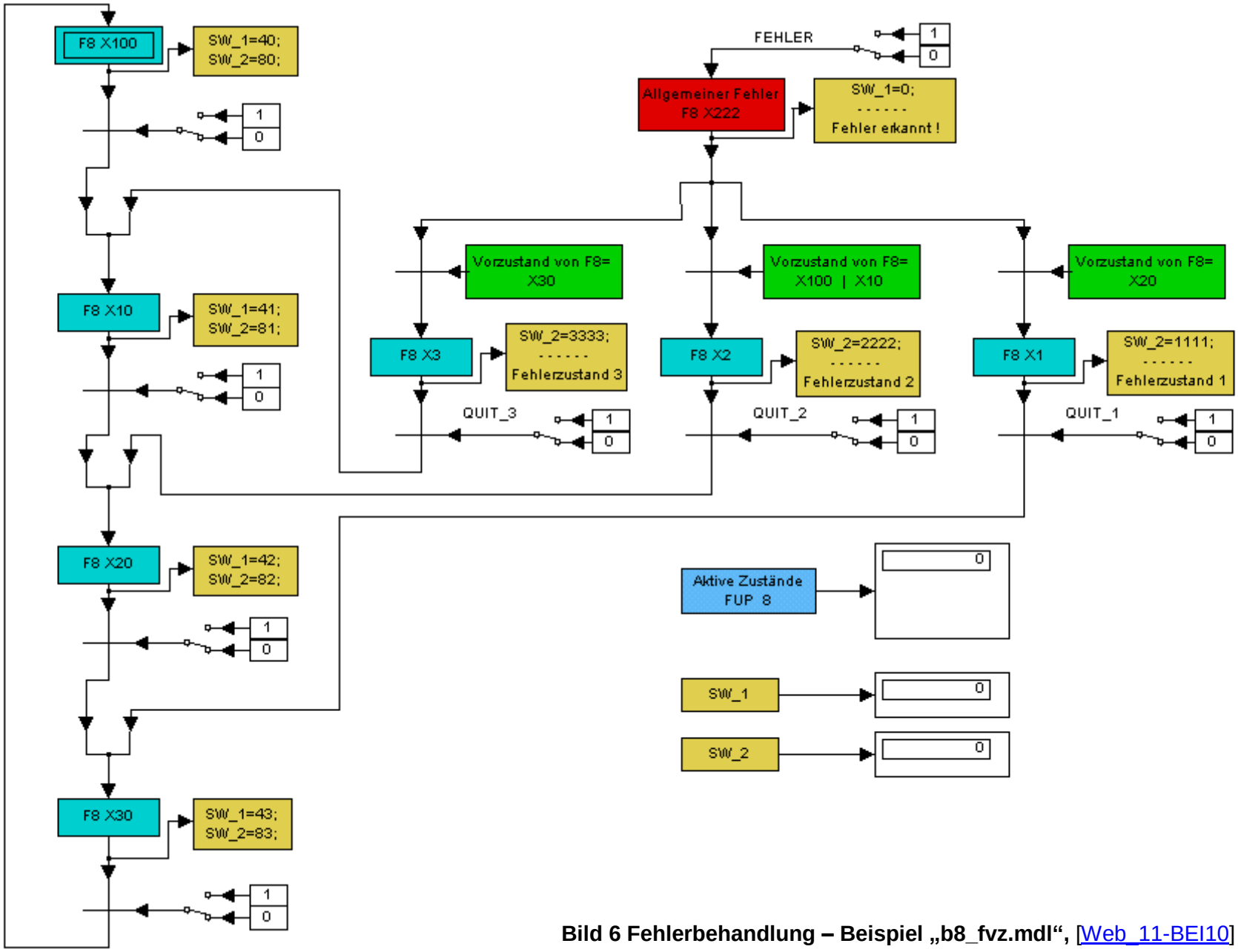


Bild 6 Fehlerbehandlung – Beispiel „b8_fvz.mdl“, [Web_11-BEI10]

```
>> Testsack
Eingabe eines Funktionsplan-Filenamens: b3_ko3t
        Hinweis: Das File enthält einen Funktionsplan mit FN = 1
        Warnung: Funktionsplan enthält keinen ZF-Block!
        Funktionsplan 1 besteht aus insgesamt 18 Zustandsblöcken
        mit Anfangszustandsblock ZA = 100
Die Anzahl der durch Reduktion eliminierten Zustandsübergänge beträgt: 0
Kompressionslauf 1: 2 Zustände eliminiert
Kompressionslauf 2: 2 Zustände eliminiert
Kompressionslauf 3: 3 Zustände eliminiert
Die Summe der durch Kompression eliminierten Zustandsblöcke beträgt: 7
ASCH=
Die Anzahl der Zustandssackgassen dieses Funktionsplanes beträgt: 1
Zustandskombinationen =
   20    22    30    32
Die Anzahl untersuchter Zustände beträgt: 414
>>
```

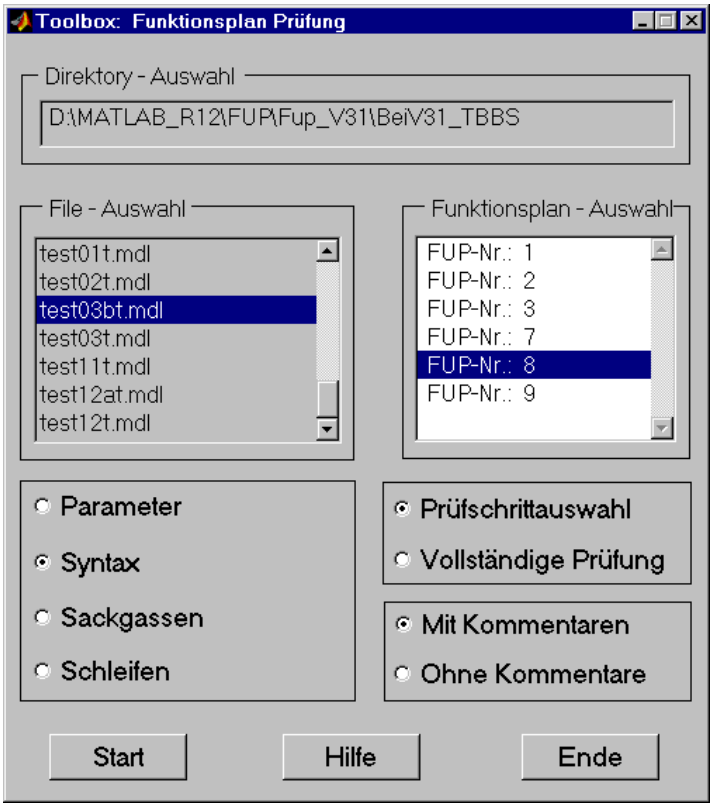
Bild 8a Ausgewählte Prüfprotokolle - Beispiel I „Sackgassentest“ [Web_4-PM3]

```
>> Testsack
Eingabe eines Funktionsplan-Filenamens: b7_efsmt
        Hinweis: Das File enthält einen Funktionsplan mit FN = 1
        Funktionsplan 1 besteht aus insgesamt 18 Zustandsblöcken
        mit Anfangszustandsblock ZA = 100 und Fehlerzustandsblock ZF = 222
Die Anzahl der durch Reduktion eliminierten Zustandsübergänge beträgt: 4
Kompressionslauf 1: 1 Zustände eliminiert
Kompressionslauf 2: 4 Zustände eliminiert
Kompressionslauf 3: 0 Zustände eliminiert
Die Summe der durch Kompression eliminierten Zustandsblöcke beträgt: 5
ASCH=
Es wurden keine Zustandssackgassen gefunden.
Die Anzahl untersuchter Zustände beträgt: 9
>>
```

Bild 8b Ausgewählte Prüfprotokolle - Beispiel II „Sackgassentest“ [Web_4-PM3]

```
>> Test_RS
Eingabe eines Funktionsplan-Filenamens: b8_fvzt
                        Hinweis: Das File enthält einen Funktionsplan mit FN = 8
        Funktionsplan 8 besteht aus insgesamt 8 Zustandsblöcken
                                                {ZA, ZX, ZE, ZF}
        mit Anfangszustandsblock ZA = 100
        und Fehlerzustandsblock ZF = 222
ASCH=1
Alle Rückführschleifen enthalten genau einen Block ZA oder ZE!
>>
```

Bild 8c Ausgewählte Prüfprotokolle - Beispiel III „Rückführschleifentest“ [Web_4-PM4]

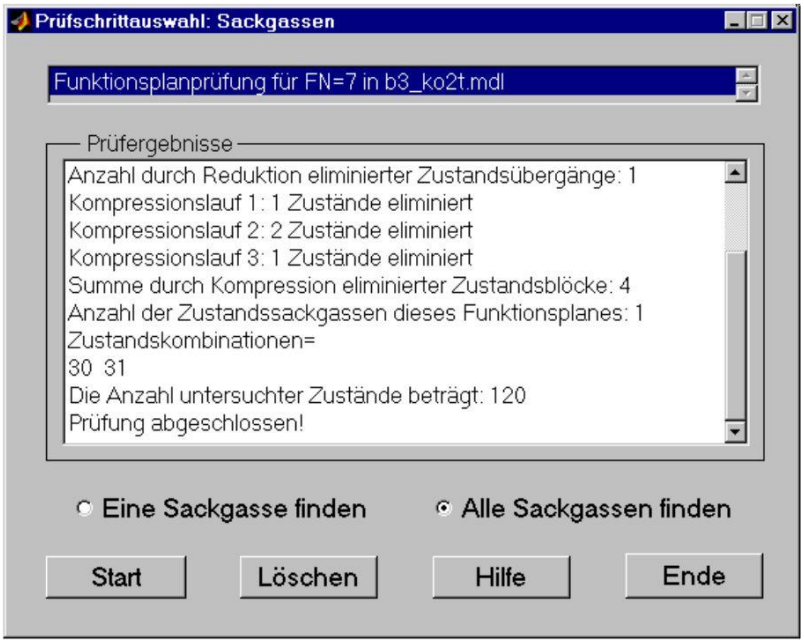


Alternative zu GUI gestützten Prüffunktionen m-File Skripte:

- TEST_PARA
- TEST_SYNT
- TEST_SACK
- TEST_WEIT

mit identischem Funktionsumfang.

GUI-Hauptfenster
Aufruf über „FupCheck“



GUI-Unterfenster für
Test auf Sackgassen
(Verklemmungen)

Bild 9 Prüfwerkzeuge für Programmierung und Test – Beispiele nach [Web 4]

Graphische Programmierung:
blockorientiert, drag & drop, beliebige hierarchische Gestaltung

Fehlerprüfwerkzeuge:
Zustands- u. FUP-Nummer, Syntax, Blockkontext, Variablenverwendung, Sackgassen

Statischer Test:
on-line Schalter zur Ereignissimulation

Dynamischer Test:
Ereignisgenerierung und Prozessmodelle, Bedienoberflächen, Arbeitsdiagramme

FN: Haben alle verwendeten Zustandsblöcke die gleiche, im GUI-Hauptfenster ausgewählte, Funktionsplan-nummer? Fehlerhafte Blöcke werden aufgelistet.

ZN: Haben alle verwendeten Zustandsblöcke des ausgewählten Funktionsplanes eine andere Zustandsnummer? Fehlerhafte Blöcke werden aufgelistet.

KB: Ist an allen Zustandsblöcken des ausgewählten Funktionsplanes ein KB-Block angeschlossen? Sind alle KB-Blöcke mit dem Ausgang eines Zustandsblocks verbunden? Fehlerhafte Blöcke werden aufgelistet.

FVZ: Ist Anschluß und Parametrierung der, im ausgewählten Funktionsplan, verwendeten FVZ-Blöcke richtig?

- Anschluß am „Enable-Eingang“ eines EX-Blocks
- Übereinstimmung von FN mit nachfolgendem Zustandsblock
- Verwendung einer FN von anderem FUP im Projekt
- Unzulässiger Anschluß an EX-Blöcke des Normalablaufs
- Unzulässiger Anschluß an EX-Blöcke nicht unmittelbar nach ZF-Block
- Verwendung von FVZ-Blöcken ohne Programmierung eines ZF-Blocks
- Fehlende FVZ-Blöcke an EX „Enable-Eingang“ nach ZF-Block

FUV: Sind alle im Projekt in FUV-Blöcken verwendeten Variablen auch in KB-Blöcken definiert? Bei Freigabe der Kommentare im GUI-Hauptfenster erfolgt eine Auflistung der vereinbarten Variablen und ein Hinweis auf mehr-fach von FUV-Blöcken empfangene Variable.

VAR in KB: Definition aller im Projekt in KB-Blöcken verwendeten Variablen auch in VAR-Blöcken? Warnung für Vari-able die nur ein einziges Mal in einem KB-Block gesetzt werden.

ZN in FVZ: Werden alle Zustandsnummern des Normalablaufes in Fehlerbehandlung verwendet?

- Auflistung der Fehlerbehandlungszustände
- Fehlende Zustände des Normalablaufs
- Falsche - zur Fehlerbehandlung gehörige - Zustandsnummern
- Nicht existente Zustandsnummern

Bild 10 Prüfmöglichkeiten – Beispiel: Details im Prüfschritt „Parameter“, [Web_4-PM1]

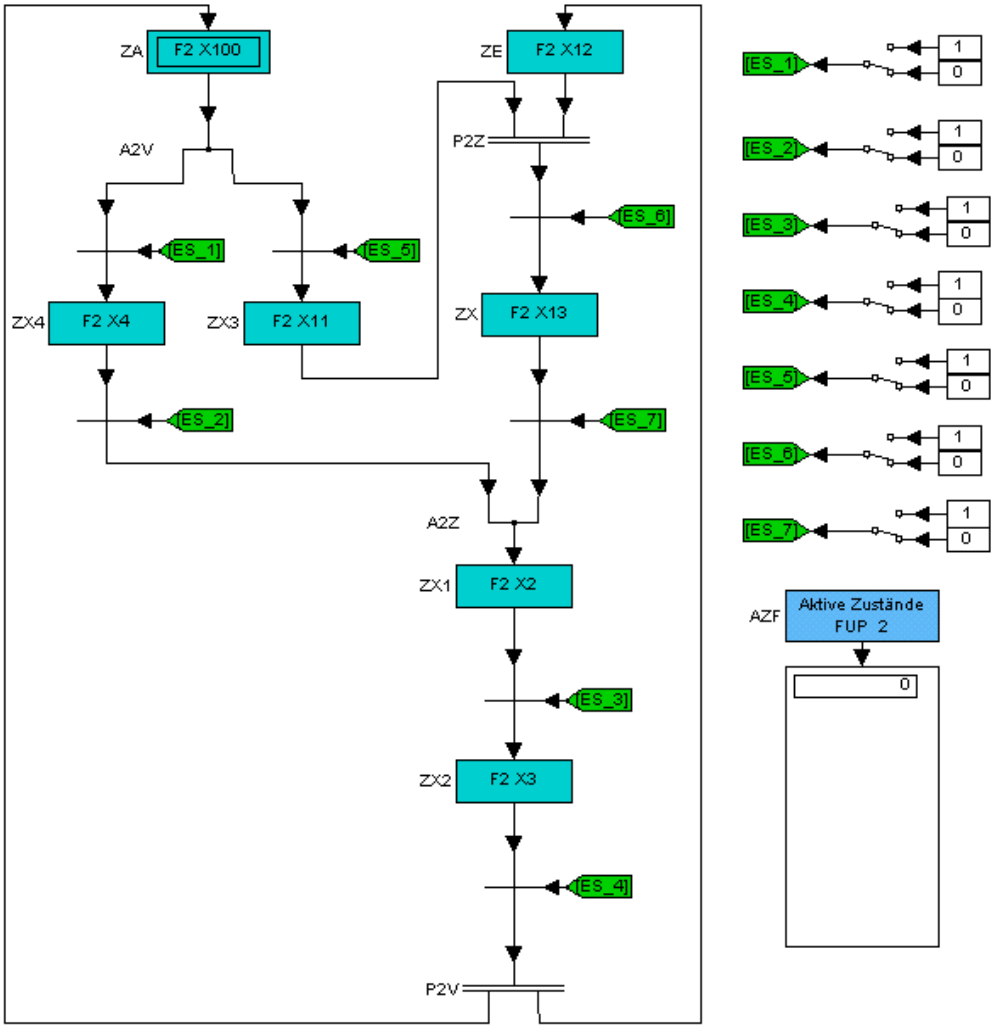


Bild 11 Statischer Test – Beispiel „b2a_pvpzt.mdl“, [Web_11-BEI4]

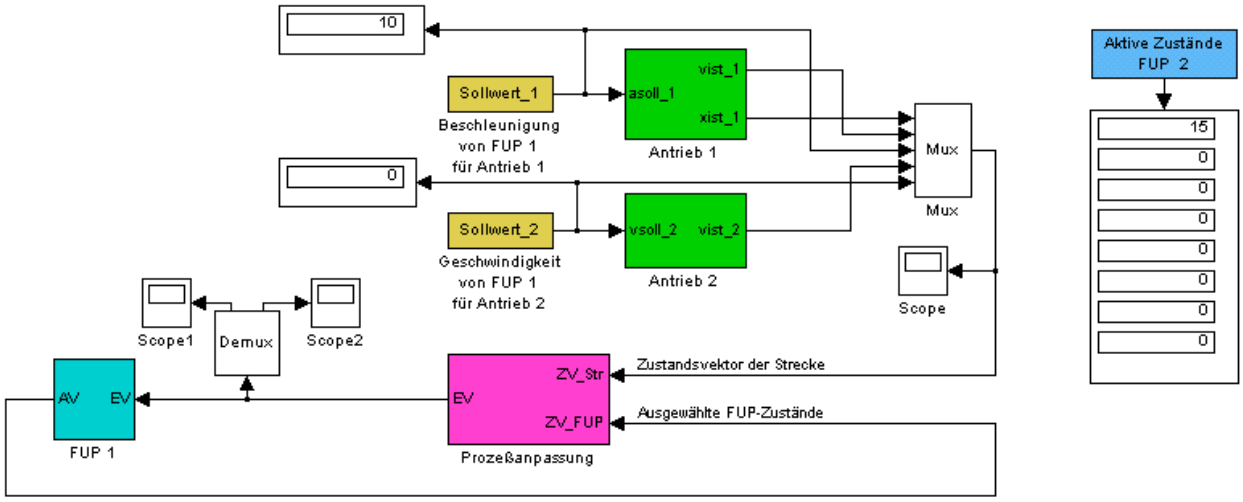


Bild 12a Dynamischer Test: Gesamtsystem - Beispiel „b5_mkb3gt.mdl“, [Web_11-Bei8] (sensitiv)
(Funktionsplan: FUP 1, Ereignisgenerierung: Prozessanpassung, Prozessmodell: Antriebe)

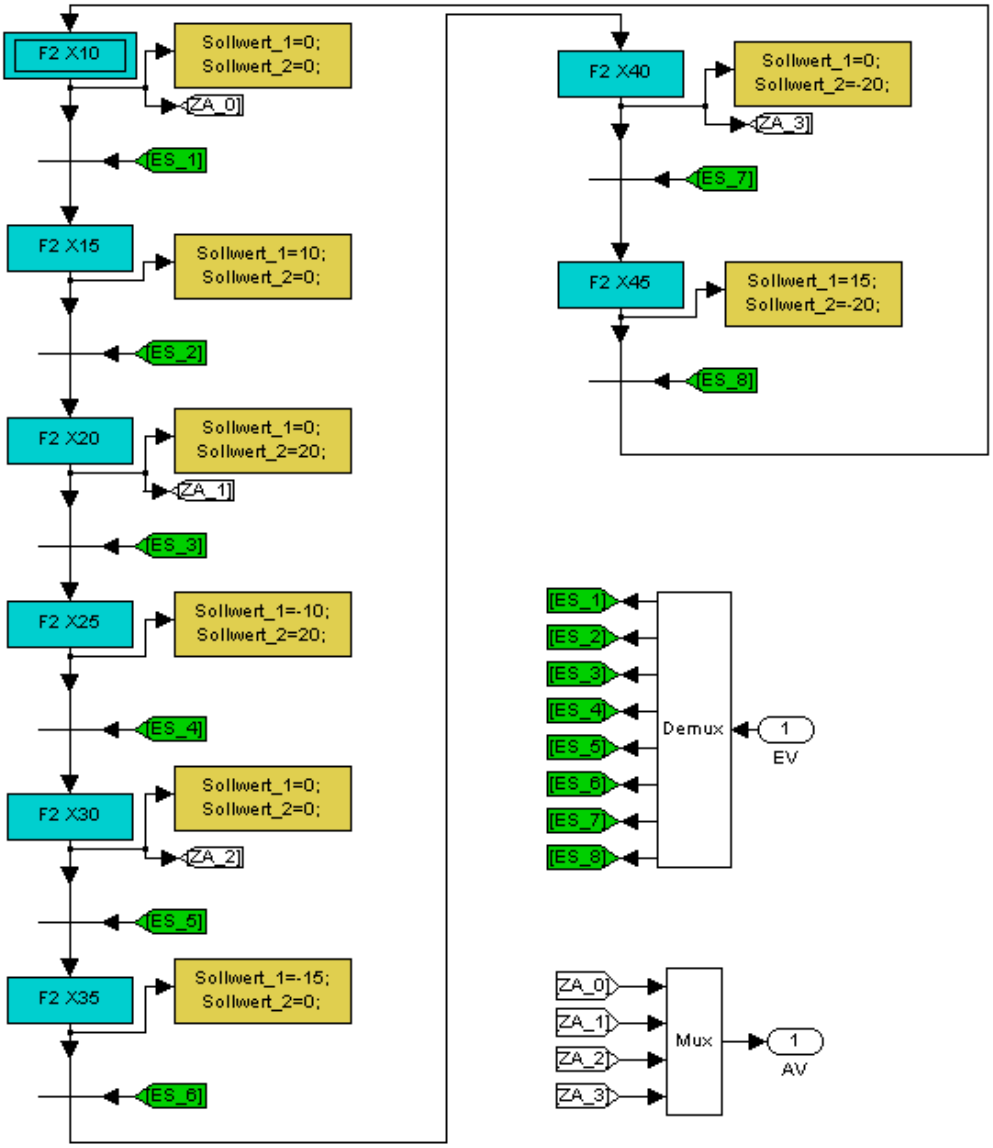


Bild 12b Dynamischer Test: Funktionsplan - Beispiel „b5_mkb3gt.mdl“, [Web_11-bei8_fup]

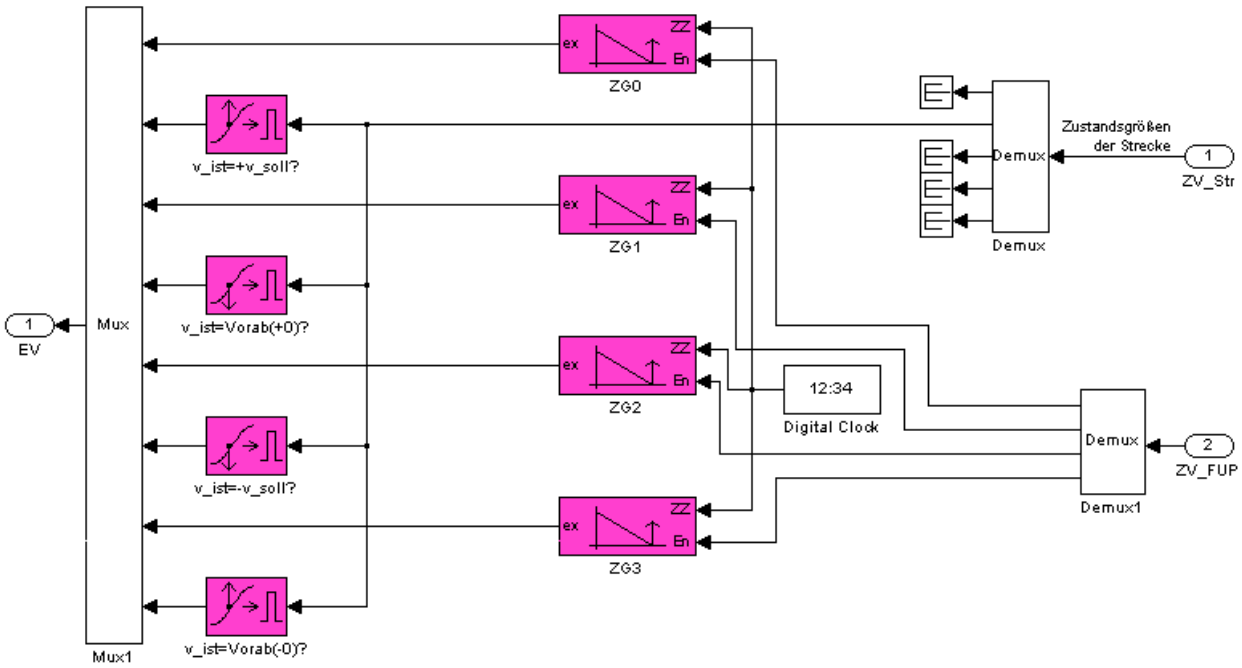


Bild 12c Dynamischer Test: Ereignisgenerierung - Beispiel „b5_mkb3gt.mdl“, [Web_11-bei8_pa]

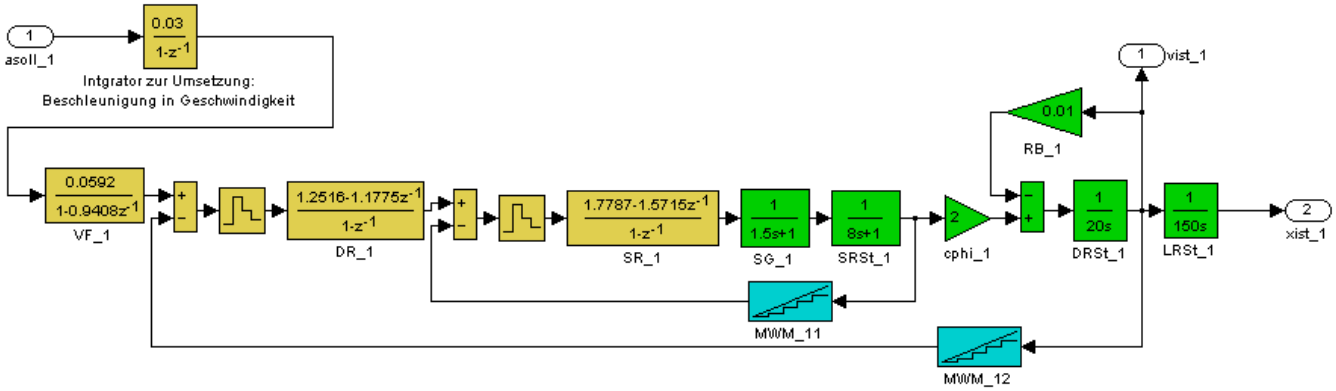
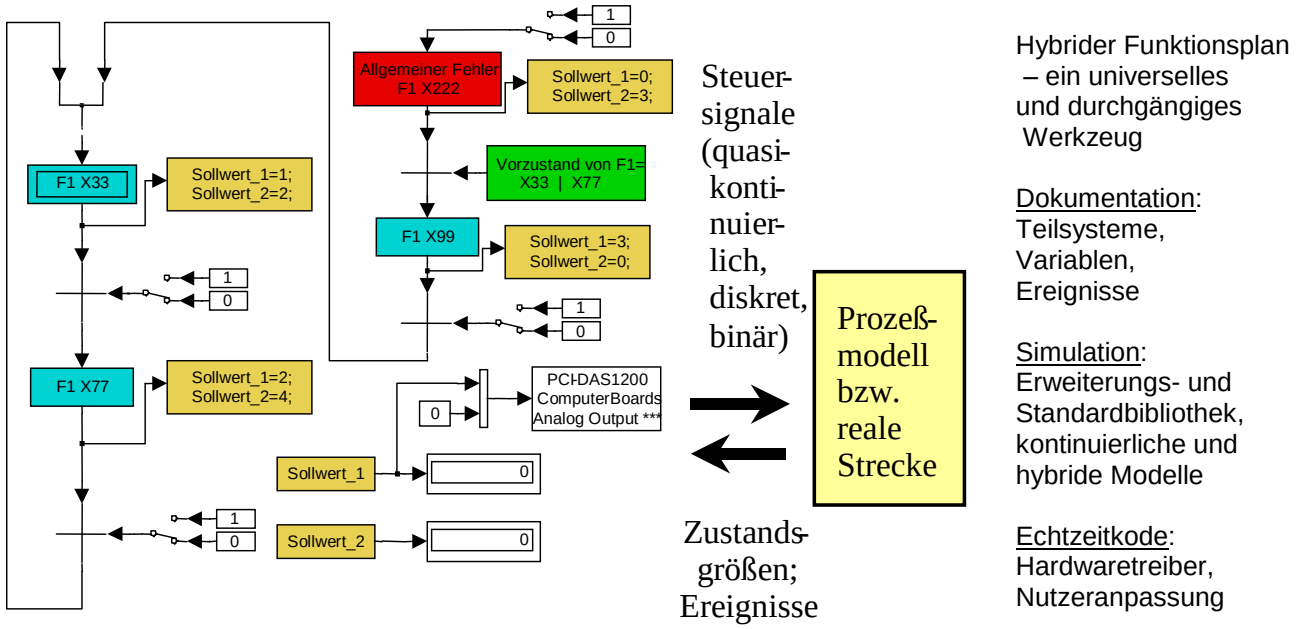


Bild 12d Dynamischer Test: Antrieb 1 - Beispiel „b5_mkb3gt.mdl“, [Web_11-bei8_a2]



Hybrider Funktionsplan – ein universelles und durchgängiges Werkzeug

Dokumentation: Teilsysteme, Variablen, Ereignisse

Simulation: Erweiterungs- und Standardbibliothek, kontinuierliche und hybride Modelle

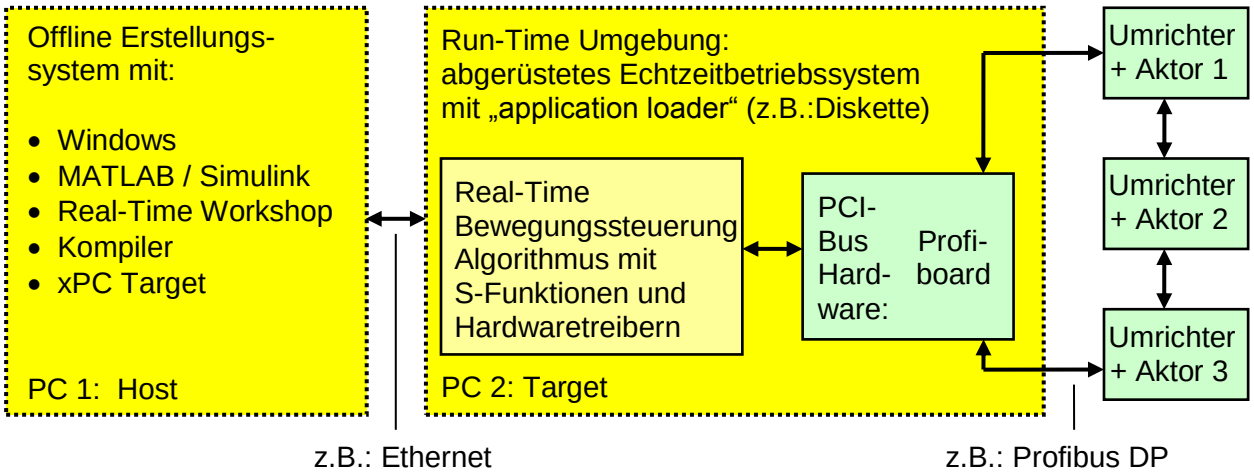
Echtzeitkode: Hardwaretreiber, Nutzeranpassung

Bild 13 Einsatz von Bibliotheken mit Hardwaretreibern

```
/* F2_NUTZER31.h                                                    */
/* Anwender-Header:   SIMULINK Erweiterungs-Blockbibliothek         */
/*                    FUP Version 3.1 Anpassung symbolischer        */
/*         Konstanten an den Anwendungsfall                         */

#define ANPAZU 199    /* maximale Anzahl paralleler Zustände        */
#define ANZVAR 499    /* maximale Anzahl von Variablen              */
#define ANZFNR   5    /* maximale Anzahl paralleler Funktions-      */
                      /* pläne:     1 <= ANZFNR <= FNRMAX           */
                      /* Achtung in Version 3.1 gilt: FNRMAX=9      */
#define REDPRU   1    /* Redundante Prüfung ausgewählter Ein-       */
                      /* gabeparameter - On-line Prüfung im         */
                      /* Echtzeitkode --> ein:1, aus:0              */
```

Bild 14 Speicher- und Laufzeitoptimierung – Quellfile F2_NUTZER31.h



Echtzeittest am realen Prozeß

Schnittstelle zum Ziel-Hardwarekompiler: „C“

Mögliche Basissoftware: Windows, UNIX, Linux, Macintosh

Hardwareplattformen – Beispiele:
PC / Laptop / PC/104 / Compact-PC inklusive FlashRAMboards, Tricore, TMS320, 683xx, …

Alternative Hardware-Schnittstellen:
Ethernet ↔ Null Modem Link;
Serieller Bus (Profibus, CAN, …) ↔ sternförmige direkte Verbindung zu E/A-Baugruppen der Antriebe (AD-Wandler, Zähler, etc.)

Bild 15 Anregungen zur Maschinenkodegenerierung am Beispiel des Einsatzes von xPC Target [Web 3]

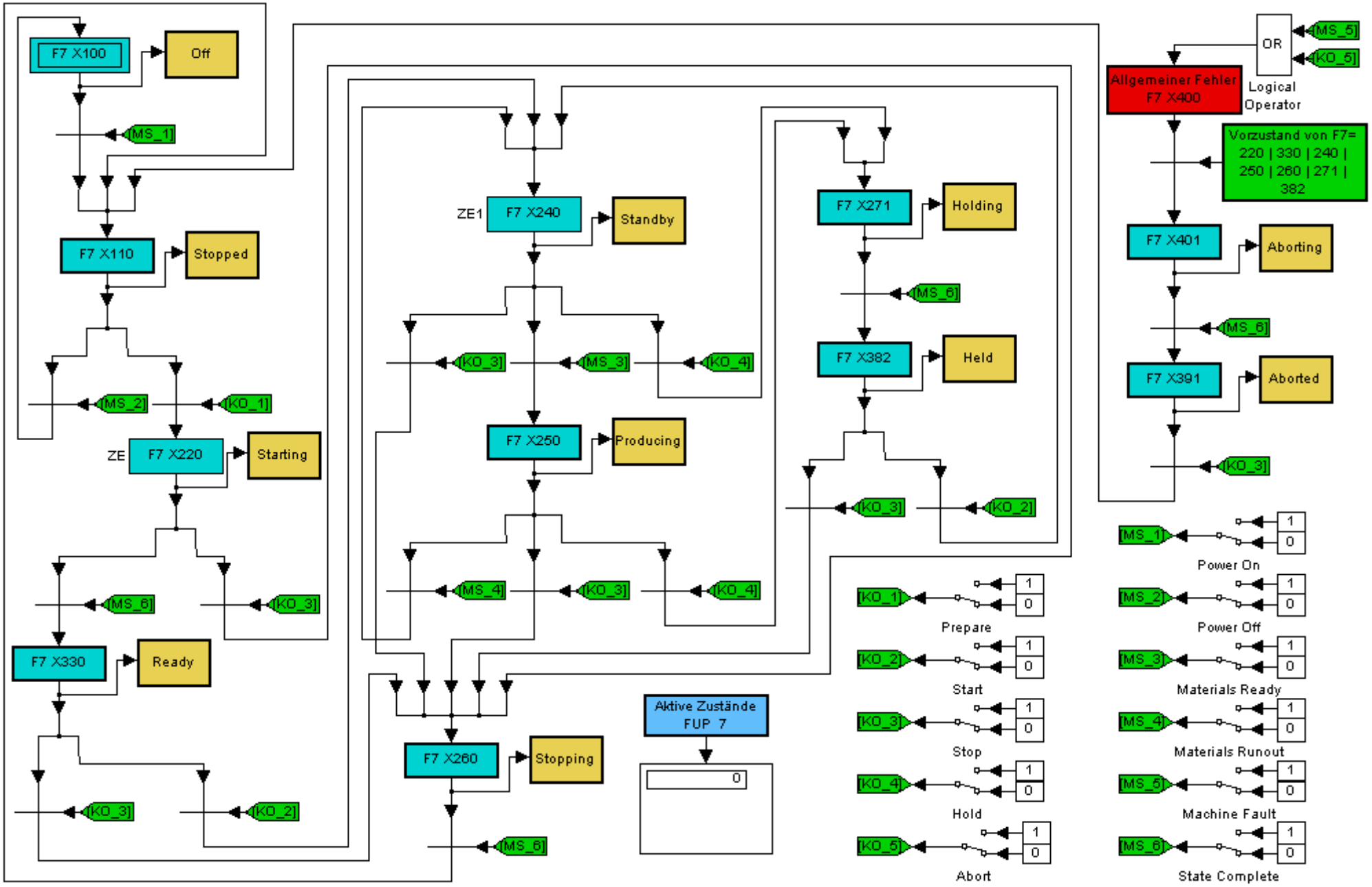


Bild 16 OMAC Maschinenmodell nach [Web_08] – Einzelzustände realisiert mit FUP

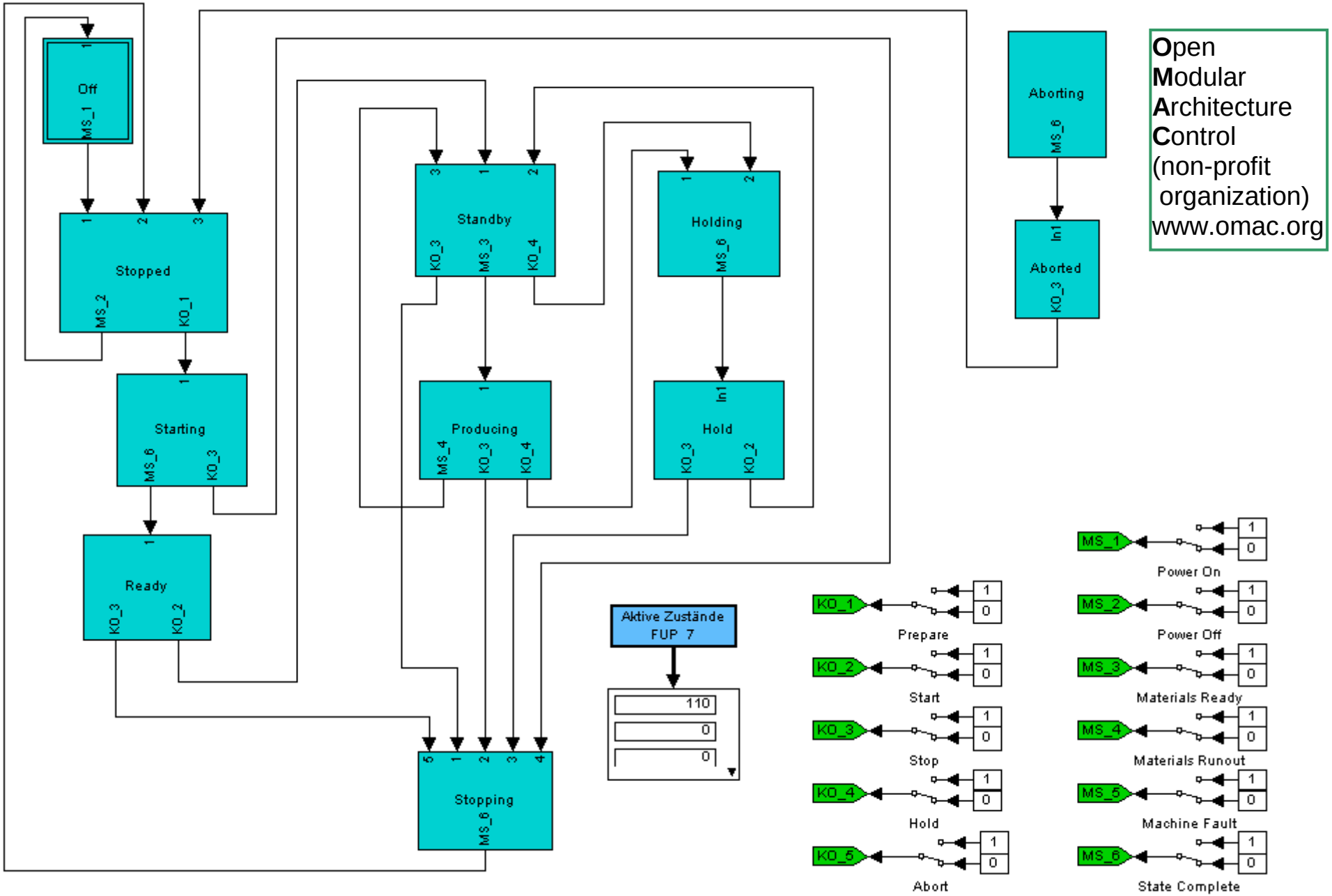


Bild 17 OMAC Maschinenmodell nach [Web_08] – Definition von Teilsystemen basierend auf Bild 16

Blockkürzel	Funktion
ZA, ZX, ZE	Anfangszustand, allgemeiner Zustand und allgemeiner Zustand mit Entkopplung gegen arithmetische Schleifen in einem Funktionsplan
EX	Ereignisblock; Definition des Angriffspunktes eines Ereignisses in einem Funktionsplan
A2V, A3V	Zwei- bzw. dreifache alternative Verzweigung des Funktionsablaufes in Abhängigkeit von zwei bzw. drei unterschiedlichen Ereignissen
A2Z, A3Z	Zwei- bzw. dreifache alternative Zusammenführung des Funktionsablaufes in Abhängigkeit von zwei bzw. drei unterschiedlichen Ereignissen
P2V, P3V	Zwei- bzw. dreifache parallele Verzweigung des Funktionsablaufes in Abhängigkeit von einem Ereignis
P2Z, P3Z	Zwei- bzw. dreifache parallele Zusammenführung des Funktionsablaufes in Abhängigkeit von zwei bzw. drei aktiven Zuständen
KB	Kommentarblock zur Vorgabe von Aktionen über String-Wertevorgaben und / oder erläuternde Kommentare zum jeweiligen Zustand
FUV	Empfang von Aktionen, die durch einen Kommentarblock einem Zustand zugeordnet wurden, zur Einspeisung in ein Prozeßmodell
ZF	Fehlerzustand in einem Prozeßmodell; Schnittstelle zur getrennten Programmierung von Normalbetrieb und Fehlerbehandlung
FVZ	Berechnung einer Booleschen Gleichung zur Ermittlung eines Fehlervorzustandes und Verzweigung der Fehlerbehandlung
ZG	Zeitgeber zur Erzeugung von definiert um tx in ms zeitverschobenen Zeitereignissen
EG_x	Ereignisgenerator; generiert bei Erreichen eines Schwellwertes einen Impuls definierter Länge der direkt als Weiterschaltereignis im Funktionsplan verwendet werden kann; x=f, s, w (fallende, steigende, wechselnd fallende und steigende Flanke)
EGE_x	Wie EG_x jedoch zusätzlich mit quasistationärem Ausgang für Boolesche Verknüpfungen der Ereignisse und Rücksetzeingang
AZF, DA	Anzeige aktiver Zustände eines Funktionsplanes bzw. ausgewählter Zustände unterschiedlicher Funktionspläne

Tabelle 1 Kurzbeschreibung der Blöcke von FUP V.3.1 (ohne ausführliche on-line Hilfetexte), [Web_10]