DGUV
Deutsche Gesetzliche Unfallversicherung
Spitzenverband

IFA Report 2/2016

# Sicherheitsbezogene Anwendungssoftware von Maschinen – Die Matrixmethode des IFA –

**IFA Report 2/2016**

# Sicherheitsbezogene Anwendungssoftware von Maschinen

## – Die Matrixmethode des IFA –

Verfasser: Michael Huelke
Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA),
Sankt Augustin

Norbert Becker, Manfred Eggeling,
Fachgebiet Automatisierungstechnik im Fachbereich
Elektrotechnik, Maschinenbau und Technikjournalismus,
Hochschule Bonn-Rhein-Sieg, Sankt Augustin

Herausgeber: Deutsche Gesetzliche Unfallversicherung e. V. (DGUV)
Glinkastr. 40
10117 Berlin
Telefon: 030 288763800
Fax: 030 288763808
Internet: www.dguv.de
E-Mail: info@dguv.de

– Juli 2016 –

Publikationsdatenbank: www.dguv.de/publikationen

Titelbild: ©Monkey Business – fotolia

ISBN (print): 978-3-86423-165-0
ISBN (online): 978-3-86423-164-3
ISSN: 2190-7986

# Kurzfassung

**Sicherheitsbezogene Anwendungssoftware von Maschinen – Die Matrixmethode des IFA**

Unternehmen im Maschinenbau realisieren Sicherheitsfunktionen immer mehr durch die Anwendungsprogrammierung von sicherheitsgerichteten Steuerungen. Die aktuellen Normen DIN EN ISO 13849 und DIN EN 62061 definieren erstmals auch Anforderungen an die Softwareentwicklung von Sicherheitsfunktionen. Dadurch sollen gefährliche systematische Fehler in der sicherheitsbezogenen Anwendungssoftware für eine Maschine vermieden werden. Wesentliche Anforderung dieser Normen ist, einen strukturierten Entwicklungsprozess einzuhalten: das V-Modell. Auch die weiteren Anforderungen zu fehlervermeidenden und -beherrschenden Maßnahmen bei der Entwicklung sind in den Normen wie üblich sehr allgemein gehalten. Zudem gibt es bislang wenige publizierte Beispiele und Vorschläge für die Umsetzung dieser Anforderungen. Daher ist die Interpretation der Normen bei der Softwareentwicklung im Maschinenbau oft unklar und bereitet Schwierigkeiten in der Umsetzung. Dies war der Anlass für ein von der DGUV gefördertes und an der Hochschule Bonn-Rhein-Sieg durchgeführtes Projekt (FF-FP0319, Laufzeit 2011 bis 2013). In dem Projekt wurde gemeinsam mit regionalen Maschinenbauunternehmen eine praktisch anwendbare Entwicklungsmethode – die Matrixmethode des IFA – hergeleitet und in einem Forschungsbericht mit vielen Beispielen dokumentiert. Dieser Forschungsbericht bildet den Kern des vorliegenden IFA Reports. Mit der hier dargestellten Matrixmethode des IFA kann Anwendungssoftware von Sicherheitsfunktionen normgerecht spezifiziert, validiert und dokumentiert werden. Darüber hinaus vermittelt der Report weitere Informationen rund um Anwendungsprogrammierung für sicherheitsbezogene Maschinensteuerungen. Der Aufwand für die Anwendungsprogrammierung ist bei Standardsteuerungen typischerweise höher als für zertifizierte Sicherheitssteuerungen. Daher beziehen sich mehrere Kapitel des Reports auf die Anwendung von Standardsteuerungen. Zur effizienten Anwendung der Matrixmethode entwickelt das IFA ein Softwaretool namens SOFTEMA. Die Beispiele des Reports sind zum Download verfügbar und können mit SOFTEMA betrachtet werden.

# Abstract

**Safety-related application software for machinery – The IFA matrix method**

Manufacturers in the machine construction sector are increasingly using application programming of safety controls in order to implement safety functions. The current EN ISO 13849 and EN 62061 standards are the first to define requirements concerning the development of software employed for safety functions. The requirements are intended to prevent hazardous systematic errors in the safety-related application software employed for a machine. The essential requirement imposed by these standards is the observance of a structured development process: the V model. The further requirements concerning measures for the avoidance and control of errors during development are also formulated in the standards in the usual very general terms. Furthermore, few examples and proposals for implementation of these requirements have been published to date. Interpretation of the standards during software development in machine construction is therefore often unclear, and presents difficulties during implementation. This situation prompted the launch of a project (FF-FP0319, project term 2011 to 2013) funded by the DGUV and conducted at the Bonn-Rhine-Sieg University of Applied Sciences. In the project, which was conducted in conjunction with machinery construction companies from the region, a development method suitable for application in the field – the IFA matrix method – was formulated and documented in a research report together with a number of examples. This research report forms the core of the present IFA Report. The IFA matrix method described here can be used to specify, validate and document the application software of safety functions in accordance with the standards. The report also provides further information on application programming for safety-related machine controls. Application programming for standard controls typically entails greater effort than for certified safety controls. Several chapters of the report therefore refer to the application of standard controls. In order for the IFA matrix method to be implemented efficiently, the IFA is developing SOFTEMA, a software tool. The examples in the report are available for download and can be viewed by means of SOFTEMA.

# Résumé

**Logiciels d'application pour machines relatifs à la sécurité – La méthode matricielle de l'IFA**

Dans le secteur de la construction mécanique, les entreprises ont de plus en plus souvent recours à des logiciels programmables de commandes liées à la sécurité pour réaliser les fonctions de sécurité. Les normes actuelles DIN EN ISO 13849 et DIN EN 62061 définissent pour la première fois également des exigences applicables à la conception des logiciels qui commandent les fonctions de sécurité, le but étant d'éviter des erreurs systématiques dangereuses dans les logiciels d'application relatifs à la sécurité des machines. L'exigence essentielle de ces normes porte sur le respect d'un processus de développement structuré : le modèle en V. Comme de coutume, les autres exigences portant sur les mesures à prendre pour éviter et maîtriser les erreurs lors du développement restent aussi très générales dans les normes. De plus, il n'existe à ce jour que peu d'exemples et de propositions publiés sur la mise en pratique de ces exigences. C'est pourquoi, lors de la conception de logiciels dans la construction mécanique, l'interprétation des normes reste souvent floue et s'avère difficile à mettre en pratique. C'est ce qui a été à l'origine d'un projet subventionné par la DGUV et réalisé par l'université des Sciences appliquées de Bonn Rhein-Sieg (FF-FP0319, durée de 2011 à 2013). Dans le cadre de ce projet, une méthode de développement facile à utiliser dans la pratique – la méthode matricielle de l'IFA – a été élaborée avec des entreprises régionales de construction mécanique, et documentée dans un rapport de recherche comportant de nombreux exemples. Ce rapport de recherche constitue l'essentiel du présent rapport de l'IFA. La méthode matricielle de l'IFA qui y est décrite permet de spécifier, de valider et de documenter les logiciels de commandes liées à la sécurité, et ce en conformité avec les normes. Le rapport fournit en outre d'autres informations concernant la programmation d'applications pour les systèmes de commande de machines relatifs à la sécurité. En règle générale, la programmation d'une application sur un système de commande standard nécessite plus de travail et de coûts que sur les automates de sécurité certifiés. C'est pourquoi plusieurs chapitres du rapport sont consacrés à l'utilisation de systèmes de commande standard. Pour utiliser efficacement la méthode matricielle de l'IFA, celui-ci a mis au point un outil logiciel baptisé SOFTEMA. Les exemples du rapport peuvent être téléchargés et visualisés avec SOFTEMA.

# Resumen

**Software de aplicación de seguridad para máquinas – El método matriz de la IFA**

Cada vez más, las empresas del ámbito de ingeniería industrial realizan funciones de seguridad mediante la programación de aplicaciones de controles relativos a la seguridad. Las normas actuales DIN EN ISO 13849 y DIN EN 62061 definen además por primera vez requisitos a tener en cuenta en el desarrollo de software para funciones de seguridad. De este modo se pretende evitar que se produzcan errores de sistema peligrosos en el software de aplicación de seguridad para una máquina. Un requisito esencial de estas normas es que se cumpla un proceso de desarrollo estructurado: el modelo en V. También el resto de los requisitos para el desarrollo con medidas para evitar y controlar fallos se describe en estas normas, como de costumbre, de manera muy general. Además, hasta la fecha se han publicado pocos ejemplos o propuestas sobre la implementación de estos requisitos. Por tanto, con frecuencia la interpretación de las normas en el desarrollo de software para maquinaria industrial resulta poco clara y genera dificultades a la hora de aplicarla en la práctica. Este es el motivo por el cual la DGUV promovió la realización de un proyecto, que se llevó a cabo en la universidad de Bonn-Rhein-Sieg (FF-FP0319, de 2011 a 2013). En este proyecto se derivó junto con empresas regionales de ingeniería industrial un método de desarrollo aplicable en la práctica, el método matriz de la IFA, que fue documentado en un informe de investigación con un gran número de ejemplos. Este informe de investigación constituye la base del presente informe de la IFA. Con el método matriz de la IFA aquí presentado se puede especificar, validar y documentar el software de aplicaciones para funciones de seguridad conforme a la norma. Además, el informe transmite otras informaciones en torno a la programación de aplicaciones para controles de maquinaria relativos a la seguridad. El trabajo que se invierte en la programación de aplicaciones es por lo general mayor para controles estándar que para controles de seguridad certificados. Por este motivo se dedican varios capítulos del informe a la aplicación de controles estándar. Para aplicar de manera eficiente el método matriz de la IFA, esta organización está desarrollando una herramienta de software denominada SOFTEMA. Los ejemplos del informe se pueden descargar en formato electrónico y se pueden consultar con SOFTEMA.

# Inhaltsverzeichnis


**1 Vorworte ..... 9**

**2 Einleitung ..... 11**
2.1 Anforderungen an Softwarequalität ..... 11
2.2 Das Forschungsprojekt FF-FP0319 der DGUV ..... 11
2.3 Der Anspruch dieses IFA Reports ..... 12

**3 Normen und Report im Überblick ..... 13**
3.1 Arten und Sprachtypen von Software ..... 13
3.2 Anforderungen an sicherheitsbezogene Anwendungssoftware (SRASW) ..... 14
3.3 Weitere informative Inhalte der DIN EN ISO 13849-1 zur SRASW ..... 15
3.3.1 Anhang G: Systematischer Ausfall ..... 15
3.3.2 Anhang J: Software ..... 15
3.4 Relevante normative Inhalte der DIN EN ISO 13849-2:2013 zur SRASW ..... 16

**4 Risikobeurteilung und Sicherheitsfunktionen ..... 17**
4.1 Risikominderung durch Sicherheitsfunktionen ..... 17
4.2 Festlegung von Sicherheitsfunktionen und deren Eigenschaften ..... 17
4.3 Einfluss der Risikobewertung auf die Softwareentwicklung ..... 17
4.4 Einfluss der Softwarestruktur auf die Softwareentwicklung ..... 18
4.5 Einfluss der Hardwarezuverlässigkeit auf die Softwareentwicklung ..... 18

**5 Fehlervermeidende Maßnahmen ..... 19**
5.1 Typischer Projektablauf ..... 19
5.2 Entwicklungsmodell V-Modell ..... 20
5.3 Beschreibung des V-Modells ..... 21
5.4 Vereinfachung des V-Modells für typische SRASW ..... 22
5.5 Dokumententypen für das vereinfachte V-Modell ..... 23
5.6 Spezifikation der Sicherheitsanforderungen und Sicherheitsfunktionen ..... 25
5.7 Programmierrichtlinien ..... 25
5.8 Modulare und strukturierte Programmierung ..... 25
5.9 Trennung von sicherheitsbezogener und nicht sicherheitsbezogener Software ..... 27
5.10 Funktionaler Test und erweiterter Test ..... 28
5.11 Testabdeckung ..... 29
5.12 Dokumentation ..... 29
5.13 Konfigurationsmanagement ..... 29
5.14 Modifikationen ..... 30
5.15 Vier-Augen-Prinzip und Unabhängigkeitsgrade ..... 30
5.16 Projektmanagement ..... 31
5.17 Externe Prüfung von SRASW ..... 32

**6 Entwicklung von sicherheitsgerichteter Anwendungssoftware ..... 33**
6.1 Matrixbasierte Spezifikation und Dokumentation ..... 33
6.2 Beispiel zur matrixbasierten Spezifikation und Dokumentation ..... 34
6.3 Spezifikation der Sicherheitsfunktionen ..... 35
6.4 Spezifikation der Steuerungshardware ..... 35
6.5 Katalog der fehlervermeidenden Maßnahmen ..... 37
6.6 Architektur des Sicherheitsprogramms und des Standardprogramms ..... 39
6.7 Softwarespezifikation mit der Cause-and-Effect-Matrix ..... 41
6.8 Verifikation und Validierung in der Matrixmethode des IFA ..... 44
6.9 Kompaktere Softwarespezifikationen ..... 46
6.10 Hinweise zur Vorverarbeitungsebene ..... 47
6.11 Berücksichtigung mehrerer Betriebsarten und eigener Funktionsbausteine ..... 48
6.12 Behandlung konfigurierbarer Sicherheitssteuerungen ..... 54
6.13 Matrixbasierte Dokumentation von eigenen Funktionsbausteinen ..... 54
6.14 Zusammenfassung der matrixbasierten Dokumentation ..... 57
6.15 Verfahren für Modifikationen ..... 58
6.16 Vereinfachung bei wiederkehrenden Sicherheitsfunktionen ..... 62
6.17 Berücksichtigung von fehlerbeherrschenden Maßnahmen ..... 64

**7 Übersicht über die behandelten Softwarebeispiele ..... 67**
7.1 Roboterfertigungszelle ..... 68
7.2 Roboterfertigungszelle mit Einrichtbetrieb ..... 69
7.3 Roboterfertigungszelle mit zusätzlicher Schutztür ..... 69
7.4 Rundtischanlage ..... 70
7.5 Werkzeugmaschine ..... 71
7.6 Safely-Limited Speed (SLS) mit Standard FU ..... 72
7.7 Safely-Limited Speed (SLS) mit Sicherheits FU ..... 73
7.8 Muting ..... 74
7.9 Zweihandbedienung ..... 74
7.10 Konfigurierbares Schaltgerät ..... 75

**8 Rolle der Embedded-Software für Anwendungsprogrammierung ..... 77**
8.1 Rolle der SRESW einer Sicherheitssteuerung ..... 77
8.2 Bewertung der SRESW einer Standardsteuerung ..... 77

**9 Einsatz von Standardsteuerungen für SRASW ..... 79**
9.1 Bestimmung der erforderlichen fehlervermeidenden Maßnahmen ..... 79
9.2 Einkanalige Architekturen ..... 79
9.3 Zweikanalige Architekturen ..... 79
9.3.1 Merkmale diversitärer SRASW ..... 79
9.3.2 Beide Kanäle mit gleicher, homogener SRASW ..... 79
9.3.3 Beide Kanäle mit diversitärer SRASW ..... 80
9.3.4 Nur ein Kanal mit SRASW ..... 80
9.4 Anwendung der Matrixmethode des IFA auf Standardkomponenten ..... 80
9.5 Einsatz von Standardkomponenten für fehlerbeherrschende Maßnahmen ..... 80

**10 Typische Test- und Überwachungsmaßnahmen in SRASW ..... 81**
10.1 Typische Techniken für Tests und Überwachungen ..... 81
10.2 Randbedingungen zu Test- und Überwachungsmaßnahmen ..... 81
10.3 Testhäufigkeit ..... 82
10.4 Weiterführende Informationen ..... 82

**11 Kombinationen mehrerer Steuerungsteile mit Software ..... 83**

**12 Validierung von SRASW ..... 85**
12.1 Allgemeine Anforderungen zur Validierung ..... 86
12.1.1 Validierung durch Analyse und Tests ..... 86
12.1.2 Validierungsplan ..... 87
12.1.3 Angaben zur Validierung ..... 87
12.1.4 Validierungsaufzeichnung ..... 87
12.2 Spezielle Anforderungen zur Validierung von SRASW ..... 87
12.2.1 Analyse der Dokumentation ..... 88
12.2.2 Test der Software ..... 88
12.3 Validierungsbeispiel aus DIN EN ISO 13849-2 Anhang E ..... 88

**13 Technische Dokumentation und Benutzerinformation ..... 86**
13.1 Technische Dokumentation ..... 89
13.2 Benutzerinformation ..... 89

**14 Das Softwaretool SOFTEMA zur Entwicklung und Prüfung von SRASW ..... 91**
14.1 Was kann SOFTEMA? ..... 91
14.2 Wie wird SOFTEMA verwendet? ..... 91
14.3 Die Benutzerschnittstelle von SOFTEMA ..... 92
14.4 Wo ist SOFTEMA zu erhalten? ..... 93
14.5 Wie wird SOFTEMA installiert und ausgeführt? ..... 93

**15 Literatur ..... 94**

**16 Abkürzungsverzeichnis ..... 96**

# 1 Vorworte

**Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)**

Mit der Einführung der Steuerungsnorm DIN EN ISO 13849-1 [1] erschien bald der BGIA-Report 2/2008 „Funktionale Sicherheit von Maschinensteuerungen“ [2], der wie sein Vorgängerreport wieder sehr stark nachgefragt wurde. Die mit 15 000 Exemplaren gedruckte deutsche Version war schnell vergriffen. Zusammen mit weiteren Hilfsmitteln zur Anwendung der Norm – die weit verbreitete Software SISTEMA und die „PLC-Drehscheibe“ – hat das IFA einen wichtigen Beitrag zur erfolgreichen Einführung von neuen Ansätzen zur Beurteilung und Auslegung der Zuverlässigkeit von elektronischen und programmierbaren Steuerungen geleistet. Dieser Ansatz mit der Betrachtung von Ausfallwahrscheinlichkeiten von Bauteilen ist in der Sicherheits-Grundnormen-Reihe DIN EN 61508 [3] verankert und mittlerweile in fast allen Industriesektoren etabliert – so auch im Maschinenbau. Die Vorgängernorm DIN EN 954-1 [4] mit ihren rein deterministischen Anforderungen wurde endgültig abgelöst. Der Performance Level ist im Maschinenbau angekommen.

Die erwähnten Reports und Hilfsmittel beschäftigen sich vorwiegend mit den zufälligen Ausfällen von Steuerungen, sprich: Bauteildefekten und Verschleiß. Aber selbst bei sehr zuverlässiger Hardware kann eine moderne Steuerung ausfallen, wenn die Anwendungssoftware, z. B. das SPS-Programm einer Sicherheits-SPS, nicht sorgfältig entwickelt wurde. Es könnte dann zu systematischen Fehlern und im Betrieb zu gefahrbringendem Versagen von Sicherheitsfunktionen kommen. Leider gab es jahrelang nur wenige veröffentlichte Beispiele und Hinweise, wie die allgemein formulierten normativen Anforderungen für Anwendungssoftware konkret und der Praxis angemessen umgesetzt werden könnten.

Daher hat das IFA schon vor Jahren begonnen, auch das Thema „Anwendungssoftware“ zu bearbeiten. Darüber hinaus wurde mit Prof. *Norbert Becker,* Fachbereich Elektrotechnik, Maschinenbau und Technikjournalismus der Hochschule Bonn-Rhein-Sieg ein geschätzter Partner gefunden, der die normative Sicht geeignet in die Praxis umsetzen kann. In den Jahren 2011 bis 2013 haben er, sein Laborteam und viele Partner aus dem Maschinenbau das von der DGUV (und damit auch von den Berufsgenossenschaften) geförderte Projekt FF-FP0319 „Normgerechte Entwicklung und Dokumentation von sicherheitsbezogener Anwendersoftware im Maschinenbau“ [5] erfolgreich bearbeitet. Es war von Anfang an beabsichtigt, den Forschungsbericht mit vielen Softwarebeispielen als zentralen Teil des vorliegenden IFA Reports zu veröffentlichen. Bis heute hat Prof. *Norbert Becker* in vielen Vorträgen und Firmenbesuchen diese Forschungsergebnisse bekannt gemacht. Damit hat er dankenswerterweise den Boden für den vorliegenden Report bereitet. Die in diesem Report sogenannte Matrixmethode des IFA ist das Ergebnis der Projektarbeit von *Becker* et al.

Dieser Report ist wie seine Vorgänger als Lehrbuch und Nachschlagewerk für Anwendungssoftware projektierende Personen gedacht. Der Report kann selbstverständlich kein Ersatz für die Anwendung der Normen sein. Die beschriebene Matrixmethode stellt aber eine aus Sicht des IFA angemessene Umsetzung der normativen Anforderungen dar. Diese Anforderungen sind für die aktuellen und wahrscheinlich auch für zukünftige Normen der funktionalen Sicherheit im Maschinensektor recht ähnlich. Daher könnte die Matrixmethode des IFA normenübergreifend erfolgreiche Anwendung finden.

Darüber hinaus enthält der Report eine Sammlung wertvoller Hinweise auf relevante Aspekte rund um sicherheitsbezogene Anwendungssoftware für Maschinensteuerungen. Im Nachgang zu diesem Report wird das IFA die kostenlose, assistierende Software „SOFTEMA“ anbieten, mit der sich auch die Beispiele im Report komfortabel betrachten und verstehen lassen. Diesem Werkzeug und der Matrixmethode des IFA wünsche ich, dass sie wie SISTEMA zu einem unabhängigen akzeptierten Standard werden: Damit Software von Maschinensteuerungen sicher und effizient spezifiziert, dokumentiert sowie validiert werden kann.

Prof. Dr. *Dietmar Reinert*
Direktor des IFA

**Hochschule Bonn-Rhein-Sieg**

Dem Vorwort von Prof. *Dietmar Reinert* ist nicht viel hinzuzufügen. Was noch bleibt, ist der Fokus auf die Historie, die industrielle Anwendung und die technische Betrachtung der Matrixmethode des IFA.

Wie ist es zur Bearbeitung des von der DGUV freundlicherweise geförderten Projektes Projekt FF-FP0319 „Normgerechte Entwicklung und Dokumentation von sicherheitsbezogener Anwendersoftware im Maschinenbau“ [5] gekommen?

Schon seit geraumer Zeit beschäftigt sich der Verfasser dieses Vorworts mit der Anwendung der modernen Sicherheitstechnik im Maschinenbau und in der verfahrenstechnischen Industrie. Die wesentlichen Punkte sind in der Vorlesung Automatisierungstechnik 2 an der Hochschule behandelt worden, weil dies als Vorbereitung der Studierenden auf die moderne Industriewelt als unerlässlich erschien. Die klassische Automatisierungstechnik und die moderne Sicherheitstechnik sind mittlerweile zusammengewachsen!

Weiterhin gibt es schon seit vielen Jahren eine enge Zusammenarbeit mit dem IFA. Insbesondere Dr. *Michael Huelke* erkannte, dass bei der industriellen Umsetzung der Anforderungen der DIN EN ISO 13849-1 [1] bzgl. der sicherheitsbezogenen SPS-Anwendersoftware noch Unterstützung sinnvoll ist. Daher ist das o. g. Förderprojekt vom IFA angeregt und letztlich vom Fachgebiet Automatisierungstechnik an der Hochschule Bonn-Rhein-Sieg bearbeitet worden.

Eine sehr wichtige Zielsetzung des Verfassers war es, das Projekt so zu bearbeiten, dass die Industrie die Ergebnisse direkt übernehmen kann. Akademische Eitelkeiten sollten bei den Resultaten überhaupt keine Rolle spielen. Beispielsweise erscheint es unrealistisch, eine Maschine als endlichen Automaten zu modellieren, um daraus Testfälle generieren zu können, obwohl die SPS-Sicherheitssoftware völlig anders implementiert wird. Aus diesem Grund ist von Anfang an beschlossen worden, an der Projektbearbeitung ortsansässige Maschinenbaufirmen in Form eines Anwenderkreises zu beteiligen, um deren bisherige Vorgehensweisen zu analysieren, industrienahe Beispiele zu akquirieren und die Ergebnisse direkt umsetzbar zu gestalten. Den Firmen Kautex Maschinenbau, Hennecke und Kuhne Anlagenbau sowie dem IFA (Dr. *Michael Huelke*) sei hier nochmals gedankt. Der VDMA, Herstellerfirmen (Siemens, Pilz, SICK) und die Berufsgenossenschaften waren über einen Lenkungsausschuss eingebunden. Der VDMA hat zusätzlich die Möglichkeit eröffnet, die Matrixmethode des IFA auf einem Anwenderworkshop am 8. November 2012 zu präsentieren. Auch dafür sei hier nochmals nachträglich gedankt. Weiterhin danke ich Herrn Dipl.-Ing. (FH) *Manfred Eggeling* für die Detailbearbeitung des Projektes.

Die Matrixmethode des IFA ist nicht völlig neu. In abgewandelten Formen wird sie bereits in der Industrie teilweise praktiziert. Es ist naheliegend, eine übersichtliche Darstellung der sicherheitsrelevanten Schaltvorgänge durch eine Matrix zu beschreiben. Damit sind sämtliche Schaltvorgänge sehr übersichtlich festgelegt. Weiterhin kann diese Beschreibung auch unmittelbar als Testgrundlage für die Validierung genommen werden. Es gibt also keine doppelte Arbeit! Gewünschte zusätzliche Testfälle lassen sich problemlos anfügen. Kann man diese Beschreibung nun auch zur Softwarespezifikation der sicherheitsbezogenen SPS-Anwendersoftware verwenden? Da sämtliche sicherheitsrelevanten Schaltvorgänge durch die Matrix beschrieben werden, ist es naheliegend, dass dies gelingt. Ein Schlüssel zur Lösung liegt darin, dass die sicherheitsbezogene SPS-Anwendersoftware sich immer in drei Teile strukturieren lässt: die Vorverarbeitungsebene, die Abschaltlogik und die Nachverarbeitungsebene. Dabei sind der erste und letzte Teil bekannt! Nur die Abschaltlogik ist unbekannt. Diese lässt sich einfach mit der Matrixmethode des IFA spezifizieren. Mit der in diesem Report dargestellten Methode lassen sich sämtliche derzeitigen und wahrscheinlich auch zukünftigen Normenanforderungen übersichtlich erfüllen. Stark vereinfacht besagen die Normanforderungen, dass der Weg von der Festlegung der Sicherheitsfunktionen bis zu deren Verifikation, Realisierung und Validierung strukturiert und im Detail nachvollziehbar sein soll. Ein wichtiger Meilenstein auf diesem Weg ist die Softwarespezifikation. Wenn dieser Meilenstein von Anwendern infrage gestellt werden sollte (man programmiert sofort!), so gibt es keine Möglichkeit, den Programmcode gegenüber den Sicherheitsfunktionen und gegenüber der Softwarespezifikation zu verifizieren (prüfen). Damit wären die Normanforderungen nicht erfüllt.

Das Förderprojekt beschreibt die Matrixmethode des IFA mittels Excel®-Tabellen in zahlreichen Beispielen. Dies ist natürlich sehr rudimentär. Das IFA stellt 2017 das kostenlos erhältliche Softwaretool SOFTEMA zur Verfügung, um diese Methode komfortabel anzuwenden. Dr. *Michael Huelke* hat SOFTEMA maßgeblich vorangetrieben und realisiert. Sämtliche Projektbeispiele stehen auch für SOFTEMA zur Verfügung.

Prof. Dr. *Norbert Becker*
Fachgebiet Automatisierungstechnik,
Fachbereich Elektrotechnik, Maschinenbau und Technikjournalismus, Hochschule Bonn-Rhein-Sieg

# 2 Einleitung

Maschinenhersteller realisieren Sicherheitsfunktionen immer mehr durch die Anwendungsprogrammierung von sicherheitsgerichteten Steuerungen. Früher definierte die DIN EN 954-1 [4] die Anforderungen an die Entwicklung von Sicherheitsfunktionen. Diese Norm gab jedoch Ende der 2000er-Jahre nicht mehr den Stand der Technik wieder und wurde von DIN EN ISO 13849-1 [1] und DIN EN 62061 [6] abgelöst, die alternativ angewendet werden können. Die neuen Normen definieren unter anderem Anforderungen an die Softwareentwicklung von Sicherheitsfunktionen. Dadurch sollen gefährliche systematische Fehler in der Anwendungssoftware für eine Maschine vermieden werden. Den Softwareentwicklern von Sicherheitsfunktionen ist die Umsetzung dieser neuen Anforderungen im Detail unklar. Dies liegt unter anderem daran, dass deren Darstellung in einer Norm naturgemäß sehr allgemein gehalten ist und es bislang nahezu keine publizierten Beispiele gab.

## 2.1 Anforderungen an Softwarequalität

Wie keine zweite Technologie übernimmt Software heute eine höhere Verantwortung als je zuvor und das trifft damit auch auf die Programmierenden zu. Als eine der wesentlichen Neuerungen in DIN EN ISO 13849-1 wurden für programmierbare Steuerungen erstmals Anforderungen an die Software und deren Entwicklung formuliert. Immerhin: Die Anforderungen in Abschnitt 4.6 der Norm ermöglichen es, sicherheitsbezogene Anwendungssoftware für Maschinensteuerungen bis zum höchsten Risiko mit dem erforderlichen Performance Level $PL_r$ e zu entwickeln – also auch für gefährliche Maschinen wie Pressen oder Schneidemaschinen.

Software ohne Fehler … gibt es in der Praxis leider nicht. Fehler in der Software entstehen nicht wie bei der Hardware durch zufällige Bauteilausfälle, sondern haben systematische Ursachen. Umso mehr muss bei der Entwicklung von sicherheitsbezogener Anwendungssoftware, die ja wie auch die Steuerungshardware zur Risikominimierung beitragen soll, alles Angemessene getan werden, um Fehler zu vermeiden. Was angemessen ist, orientiert sich einerseits am erforderlichen Performance Level $PL_r$ und damit am vorliegenden Risiko. Andererseits ist bekannt, in welchen Phasen der Softwareentwicklung sich sicherheitskritische Fehler bevorzugt und mit besonders gravierender Wirkung einschleichen und solange unentdeckt bleiben, bis sie beim Betrieb der Maschine zum Ausfall führen. Gemeint sind im Lebenszyklus einer Maschine die Phasen Spezifikation, Softwareentwurf und im Laufe ihrer langen Betriebsdauer die Modifikationen. Daher zielen die Anforderungen der DIN EN ISO 13849-1 – und die hier vorgestellte Matrixmethode des IFA – besonders auf die Fehlervermeidung in diesen Phasen. Leider werden in der Praxis gerade diese Phasen der Anwendungsprogrammierung oft mit weniger Aufmerksamkeit bedacht.

## 2.2 Das Forschungsprojekt FF-FP0319 der DGUV

Im Projekt FF-FP0319 „Normgerechte Entwicklung und Dokumentation von sicherheitsbezogener Anwendersoftware im Maschinenbau“ der DGUV [5] (2011 bis 2013) wurden vom Projektnehmer, Prof. *Norbert Becker* (Hochschule Bonn-Rhein-Sieg), mehrere konkrete Vorgehensweisen für die Umsetzung der in den neuen Normen enthaltenen Anforderungen an die Softwareentwicklung von Sicherheitsfunktionen für Maschinen erarbeitet und anhand von industriellen Beispielen evaluiert und dokumentiert. Ziel war es, sowohl die Vorgehensweisen als auch deren Anwendungen in einem Forschungsbericht zu beschreiben, den das IFA anschließend als Teil des vorliegenden IFA Reports der Öffentlichkeit zur Verfügung stellt.

Zur Evaluierung der Projektergebnisse wurden während der Projektlaufzeit zwei Gremien eingerichtet:

- ein Anwenderkreis bestehend aus lokalen Industrieunternehmen (Kautex Maschinenbau, Hennecke, Kuhne Anlagenbau), dem IFA und der TÜV Rheinland Akademie,
- der Forschungsbegleitkreis mit Mitgliedern von Steuerungsherstellern (Pilz, SICK, Siemens), Berufsgenossenschaften, IFA, Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA), TÜV Rheinland Akademie, Kommission Arbeitsschutz und Normung (KAN) und Anwendern.
- Zusätzlich wurde die Methode in verschiedenen Industrieunternehmen vorgestellt und diskutiert.

Das Projekt gliederte sich in die Aufgaben

- Entwicklung einer Methode und
- anschließende Vorstellung der Methode im und Bewertung durch den Anwenderkreis und den Forschungsbegleitkreis.

Dabei wurden mehrere Methoden zur Spezifikation von Anwendungssoftware untersucht:

- Beschreibung der Anwendungssoftware als Zustandsautomat,
- Spezifikation über Checklisten,
- Spezifikation mit Tabellen/Matrizen.

Beschreibt man die Anwendungssoftware einer realen Maschine als Zustandsautomat [7], wobei sämtliche Betriebsarten berücksichtigt werden, so ist dies im Allgemeinen sehr komplex. Weiterhin erfolgt die spätere Programmierung der Anwendungssoftware völlig anders als die Darstellung als Zustandsautomat in einer grafischen oder textorientierten Programmiersprache. Dies gilt insbesondere für die Sicherheitssoftware, in der die

Verwendung von zertifizierten Funktionsbausteinen üblich ist. Im Maschinenbau ist die Vorgehensweise mit Automatenbeschreibungen unüblich. Zustandsautomaten werden dagegen genutzt bei der Spezifikation von komplexen sicherheitsrelevanten Funktionsbausteinen (Bibliotheksbausteinen) [8], was aber nicht das primäre Thema dieses Forschungsprojektes war.

Weiterhin wurde eine auf Checklisten basierte Methode entwickelt. Die Sicherheitsfunktionen werden hier durch checklistenorientierte Formulare beschrieben, die im Laufe der weiteren Spezifikation immer weiter verfeinert werden. Nach den Vorstellungen im Anwenderkreis und im Forschungsbegleitkreis war schnell klar, dass die checklistenbasierte Methode zur Entwicklung und Dokumentation von Sicherheitssoftware ebenfalls nicht industrietauglich ist. Aber: Viele Unternehmen dokumentieren und spezifizieren bereits die Sicherheitssoftware in Form von Tabellen. Dadurch gestützt wurde eine matrixbasierte Form der Spezifikation und Dokumentation von Sicherheitssoftware entwickelt. Diese stieß auf viel mehr Akzeptanz bei Vorstellungen in der Industrie.

Dieses im Folgenden „Matrixmethode des IFA“ genannte Verfahren haben der Anwenderkreis und der Forschungsbegleitkreis positiv aufgenommen. Aus den Diskussionen ergaben sich zahlreiche Verbesserungen an der Darstellung. Verschiedene Beispiele wurden in dieser Darstellungsform erarbeitet, um möglichst viele praxisrelevante Fälle zu beschreiben. Zusätzlich ist ein größeres Beispiel einer Werkzeugmaschine umgesetzt worden, um zu zeigen, dass auch größere Anlagen mit dieser Matrixmethode des IFA beschrieben werden können.

Die Matrixmethode des IFA wurde als Zwischenergebnis des Projektes beim VDMA-Workshop „Funktionale Sicherheit – Sichere Anwendersoftware im Maschinenbau“ am 8. November 2012 in Frankfurt am Main der Öffentlichkeit vorgestellt. Zusätzlich gab es danach vom Projektteam noch mehrere Publikationen [9] und viele Vorstellungen bei Firmen, die weitgehend positiv verliefen und zu weiteren Anregungen geführt haben.

## 2.3 Der Anspruch dieses IFA Reports

Mit der hier dargestellten Matrixmethode des IFA kann die normgerechte Spezifikation und Dokumentation der Anwendungssoftware von Sicherheitsfunktionen umgesetzt werden. Wenn diese beispielhaft gezeigte Vorgehensweise eingehalten wird, kann man davon ausgehen, dass die relevanten Anforderungen der DIN EN ISO 13849-1 an die sicherheitsbezogene Anwendungssoftware (Abschnitt 4.6) der Sicherheitsfunktionen erfüllt sind.

Außer dieser Vorgehensweise gibt es sicherlich auch andere Methoden, mit denen die Anforderungen gleichwertig erfüllt werden können. Daher beansprucht die Matrixmethode des IFA nicht, exklusiv die Normenanforderungen erfüllen zu können.

Der Aufwand bei der Anwendungsprogrammierung ist bei zertifizierten Sicherheitssteuerungen typischerweise geringer als bei Standardsteuerungen. Mehrere Abschnitte dieses Reports beziehen sich nur auf die Anwendung von Standardsteuerungen und sind entsprechend gekennzeichnet.

Zusätzlich zur Anwendung der Matrixmethode müssen im Einzelfall weitere Details spezifiziert und geprüft werden, z. B.

- herstellerspezifische Parametrierungen der verwendeten Peripheriegeräte (z. B. Umrichter, Sensoren),
- weitere sicherheitsrelevante Funktionen der Maschinensteuerungen, die nicht von der Matrixmethode des IFA abgedeckt sind, und
- zusätzliche sonstige, nicht direkt sicherheitsrelevante Funktionen, wie z. B. spezielle Quittierphilosophien.

# 3 Normen und Report im Überblick

Das DGUV Projekt FF-FP0319 [5] und dieser Report behandeln die Umsetzung der Anforderungen aus der im Maschinensektor hauptsächlich angewendeten Normenreihe DIN EN ISO 13849, bestehend aus zwei Teilen [1; 10]. Es gibt aber weitere anwendbare Normen, von denen die Norm DIN EN 62061 [6] ebenfalls unter der Maschinenrichtlinie harmonisiert ist. DIN EN 62061 ist zwar auf elektrische, elektronische und programmierbar elektronische Systeme beschränkt und daher für viele Maschinen mit hydraulischen und pneumatischen Steuerungsteilen nur eingeschränkt geeignet; aber in Bezug auf Anwendungssoftware werden in DIN EN 62061 in Abschnitt 6.11.3 ähnliche Anforderungen und Vorgehensweisen wie in DIN EN ISO 13849-1 formuliert. Beide Normen haben für Anwendungssoftware den vergleichbaren Anwendungsbereich für Steuerungen bis hin zur höchsten Sicherheitsstufe für den Maschinensektor (bis zu PL e bzw. SIL 3). Die Gremien beider Normen haben inzwischen auch die Gleichwertigkeit der Anforderungen untersucht und in dem gemeinsamen Report DIN ISO/TR 23849:2010 [11] dokumentiert. Im Detail sind die Anforderungen unterschiedlich ausgeprägt: DIN EN 62061 als Sektornorm der Normenserie DIN EN 61508 [3] beschreibt z. B. den Aspekt des „Managements der funktionalen Sicherheit“ sehr ausführlich.

Eine Empfehlung der in diesem Report beschriebenen Vorgehensweise spricht das IFA ausdrücklich nur aus in Bezug auf die Erfüllung der Anforderungen der DIN EN ISO 13849 der ersten Fassung und der Änderung 1 dieser Norm. Gleichwohl mag diese Vorgehensweise annähernd auch für DIN EN 62061 geeignet sein.

## 3.1 Arten und Sprachtypen von Software

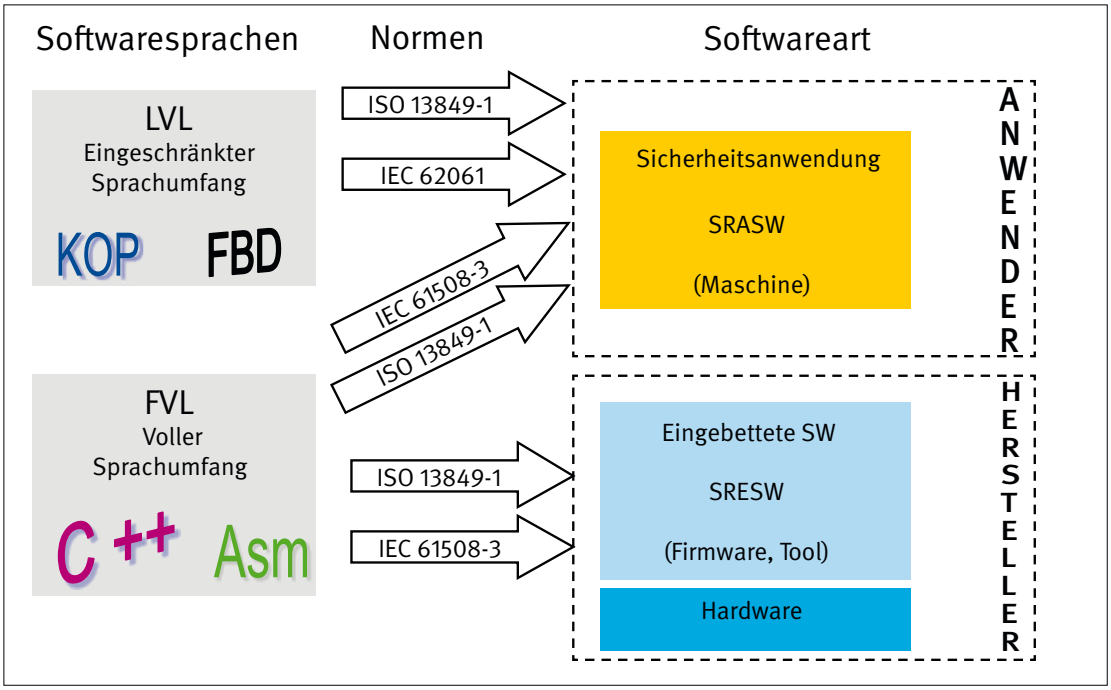
Sicherheitsbezogene Software ist ein bestimmter Anteil der Software eines Systems (hier: einer Maschine wie in Abbildung 1 rechts), die zur Ausführung von sicherheitsbezogenen Steuerungsfunktionen (hier: Sicherheitsfunktionen) in einem sicherheitsbezogenen System (hier: Steuerungsteile einer Maschine) verwendet wird. Diese im Englischen als „safety-related software“ bezeichnete Software kennt nun mehrere Ausprägungen hinsichtlich der Softwaresprachen und der Softwareart. Hierbei stimmen die Definitionen von DIN EN (ISO) 13849 und 62061 noch weitgehend überein.

In den Normen werden zwei Typen von Softwaresprachen definiert:

- FVL: Programmiersprache mit nicht eingeschränktem Sprachumfang (englisch: Full Variability Language),
  siehe DIN EN ISO 13849-1 [1], Abschnitt 3.1.35

  Das ist ein Typ einer Sprache mit der Fähigkeit, einen großen Bereich von Funktionen zu implementieren, Beispiele: C, C++, Assembler. Ein typisches Beispiel von Systemen für die Verwendung von FVL sind Embedded-Systeme (siehe unten). Im Bereich der Maschinen wird FVL in Embedded-Software und gelegentlich in Anwendungssoftware eingesetzt.

Abbildung 1:
Zusammenhang zwischen Softwaresprachen, Softwarearten und anzuwendenden Normen, Asm = Assembler



- LVL: Programmiersprache mit eingeschränktem Sprachumfang (englisch: Limited Variability Language), siehe DIN EN ISO 13849-1 [1], Abschnitt 3.1.34

  Das ist ein Typ einer Sprache mit der Fähigkeit, vordefinierte, anwendungsspezifische Bibliotheksfunktionen zu kombinieren, um die Spezifikation der Sicherheitsanforderungen zu implementieren. Typische Beispiele von LVL (Kontaktplan, Funktions-Blockdiagramm) sind in DIN EN 61131-3 [2], die Norm für SPS, angegeben. Ein typisches Beispiel für ein System, das LVL verwendet, ist die speicherprogrammierbare Steuerung (SPS).

Bei den Softwarearten unterscheidet man:

- SRESW: Sicherheitsbezogene Embedded-Software (englisch: safety related embedded software), siehe DIN EN ISO 13849-1 [1], Abschnitt 3.1.37

  Das ist eine Software, die als Teil des Systems durch den Steuerungshersteller geliefert wird und die durch den Anwender der Maschine nicht verändert werden kann. Üblicherweise wird SRESW in FVL geschrieben. Typische Beispiele sind Firmware, Betriebssystem, Laufzeitsystem etc.

- SRASW: Sicherheitsbezogene Anwendungssoftware (englisch: safety related application software), siehe DIN EN ISO 13849-1 [1], Abschnitt 3.1.36

  Das ist eine Software, die speziell für die Anwendung vom Hersteller in die Maschine implementiert wird. Sie enthält üblicherweise logische Sequenzen, Grenzwerte und Ausdrücke zum Verarbeiten und Steuern der entsprechenden Eingänge, Ausgänge, Berechnungen und Entscheidungen, um die notwendigen Anforderungen des sicherheitsbezogenes Teil einer Steuerung zu erfüllen. Typisches Beispiel ist das SPS-Programm einer Sicherheits-SPS. Üblicherweise wird SRASW in LVL geschrieben.

- Parametrier-Software: Einige sicherheitsbezogene Steuerungsteile benötigen in ihrer Anwendung noch zusätzliche Parametrierung: z. B. Frequenzumrichter mit integrierter Drehzahlüberwachung (SLS), bei denen die Überwachungsdrehzahl einzugeben ist. Die Parametrierung erfolgt durch den Steuerungsanwender üblicherweise mit einer dedizierten Parametrier-Software, die vom Lieferanten des Steuerungsteils bereitgestellt wird. Diese Parametrier-Software erzeugt auch eine eigene Dokumentation.

  Sicherheitsbezogene Parameter können aber auch über projektspezifische Lösungen, z. B. ein Standard-Bediengerät, eingegeben werden. In diesen Fällen muss allerdings der Steuerungsanwender selbst eine fehlerfreie Parametrierung sicherstellen. In diesen Fällen sind normative Anforderungen nach DIN EN ISO 13849-1 [1], Abschnitt 4.6.4, zu erfüllen.

Die anzuwendenden Normen und damit die Anforderungen an die Softwareentwicklung richten sich nach dem verwendeten Typ der Softwaresprache (LVL oder FVL) und der Softwareart (SRASW oder SRESW), siehe Abbildung 1.

Dieser Report konzentriert sich auf Anwendungssoftware, die üblicherweise in einer SPS-Sprache programmiert wird und dem Sprachtyp LVL entspricht (Abbildung 1 oben). Damit sind die Anforderungen aus DIN EN ISO 13849-1, Abschnitt 4.6.3, relevant. In diesem Report wird daher im Folgenden für diese Anwendungssoftware die normative Abkürzung SRASW verwendet.

SRASW, die in FVL programmiert wird, z. B. in C, wird aufgrund der höheren Wahrscheinlichkeit von systematischen Fehlern nach der Norm wie Embedded-Software (Abschnitt 4.6.2 der DIN EN ISO 13849-1) betrachtet.

Als Entwicklungsmodell ist sowohl für SRESW als auch für SRASW das vereinfachte V-Modell aus Abschnitt 4.6.1 der Norm anzuwenden (siehe Abschnitt 5.2 dieses Reports).

## 3.2 Anforderungen an sicherheitsbezogene Anwendungssoftware (SRASW)

Nachdem in Abschnitt 4.6.1 der DIN EN ISO 13849-1 der Entwicklungsprozess skizziert ist, werden für SRASW im Abschnitt 4.6.3 die normativen Anforderungen an die Software selbst, an die benutzten Entwicklungswerkzeuge sowie an die Entwicklungsaktivitäten beschrieben. Diese Anforderungen tragen ebenfalls zur Fehlervermeidung bei. Der dazu erforderliche Aufwand soll – ähnlich wie bei der Hardware des programmierbaren SRP/CS – der jeweils notwendigen Risikominderung entsprechend angemessen sein. Daher werden die Anforderungen bzw. deren Wirksamkeit mit zunehmendem $PL_r$ der realisierten Sicherheitsfunktion(en) sinnvoll gesteigert. DIN EN ISO 13849-1 nennt also keine Maximalanforderungen, die für jede Software – unabhängig vom $PL_r$ – notwendig wären. Abbildung 2 zeigt, dass es bei SRASW (so wie auch bei SRESW) für alle $PL_r$ zunächst ein geeignetes Bündel von Basismaßnahmen gibt. Folgende in Kapitel 5 beschriebene Basismaßnahmen sind schon für die Entwicklung von Software für $PL_r$ a oder b gefordert:

- Entwicklungslebenszyklus mit Verifikation und Validierung (Abschnitt 5.2),
- Spezifikation der Sicherheitsanforderungen (Abschnitt 5.6),
- Dokumentation von Spezifikation und Entwurf (Abschnitt 5.12),
- modulare und strukturierte Programmierung (Abschnitt 5.7),
- funktionale Tests (Abschnitt 5.10),
- geeignete Entwicklungsaktivitäten nach Änderungen (Abschnitt 5.14).

Für Software, die für $PL_r$ c bis e eingesetzt wird, gelten zusätzlich zu den Basismaßnahmen weitere fehlervermeidende Maßnahmen. Letztere sind für $PL_r$ c mit geringerer Wirksamkeit, für $PL_r$ d mit mittlerer Wirksamkeit und für $PL_r$ e mit höherer Wirksamkeit

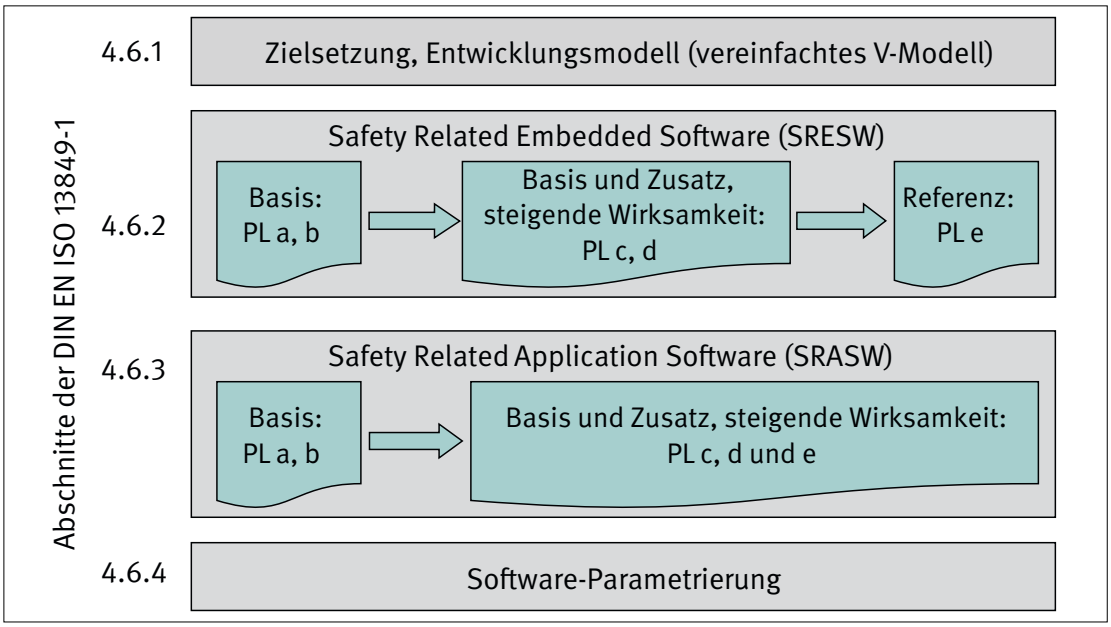
gefordert. Unabhängig davon, ob die Software nur in einem oder in beiden Kanälen einer beliebigen Kategorie mitwirkt: Als Maßstab für die Anforderungen gilt immer der $PL_r$ der realisierten Sicherheitsfunktion(en), siehe Abschnitt 4.3.

Der Aspekt „steigende Wirksamkeit“ bezieht sich auf den zunehmenden Effekt der Fehlervermeidung. Dies soll hier an der wichtigen Aktivität der „Spezifikation“ illustriert werden. So kann es z. B. für $PL_r$ c ausreichend sein, wenn die Programmierenden die Spezifikation selbst verfassen und sie später selbst gegenlesen. Soll aber die gleiche Software für $PL_r$ e eingesetzt werden, so muss ein höherer Grad der Fehlervermeidung erreicht werden. Dann kann es notwendig sein, dass nicht Programmierende selbst die Spezifikation schreiben, sondern z. B. die „Projektleitung Software“. Darüber hinaus könnte das Review dieser Spezifikation gemeinsam vom Programmierenden und einer unabhängigen Person, z. B. dem Hardware-Projektierenden, durchgeführt werden. Mehr Personen sehen (meist) mehr Fehler.

Über die mehr oder weniger wirksamen Ausprägungen von Anforderungen gibt es zum Zeitpunkt des Erscheinens dieses Reports leider keine bekannte Literatur. So ist jeder Normenanwendende selbst in der Pflicht, die Ausprägungen von Anforderungen festzulegen.

Die hier zitierten normativen Anforderungen werden im Rahmen der Matrixmethode des IFA in einem separaten Dokument „A4 – Anforderungen“ (siehe Abschnitt 5.5) dargestellt und projektbezogen kommentiert.

Abbildung 2:
Abstufung der Anforderungen an sicherheitsbezogene Software (DIN EN ISO 13849-1)




## 3.3 Weitere informative Inhalte der DIN EN ISO 13849-1 zur SRASW

Neben den o. g. normativen Anforderungen enthält die Norm im Teil 1 weitere relevante Inhalte in den informativen Anhängen G und J. „Informativ“ bedeutet hier: Sie sind Erläuterungen zum besseren Verständnis des teils abstrakten Normentextes und spiegeln die Vorstellungen der Normensetzer wider – eine Anleitung zur typischen Umsetzung, die als Stand der Technik anzusehen ist.

### 3.3.1 Anhang G: Systematischer Ausfall

Fehler in SRASW können zu systematischen Ausfällen führen. Der informative Anhang G betrachtet diese Ausfallart, bezieht sich aber im Wesentlichen auf Steuerungshardware, da ja schon

im normativen Abschnitt 4.6 alles zur Steuerungssoftware beschrieben ist.

In diesem Anhang wird unterschieden zwischen Vermeidung und Beherrschung systematischer Ausfälle. In Abschnitt G.2 wird als ausfallbeherrschende Maßnahme genannt, dass eine Programmablaufüberwachung verwendet werden muss, um fehlerhafte Programmabläufe zu erkennen. Bei einer zertifizierten Sicherheits-SPS wird diese Überwachung typischerweise in der Firmware vorhanden sein, um z. B. einen fehlerhaften Ablauf der SRASW zu erkennen. Bei einer Standardsteuerung müsste diese Maßnahme in der SRASW selbst zuverlässig realisiert werden.

Als fehlervermeidende Maßnahme speziell bei der Integration der Steuerung wird nur auf die Standardmaßnahmen Funktionstest, Projektmanagement und Dokumentation hingewiesen (siehe auch Kapitel 5 dieses Reports).

### 3.3.2 Anhang J: Software

Der informative Anhang J demonstriert im Bild J.1 zunächst anhand eines Softwarebeispiels sehr anschaulich den modularisierten Aufbau mit Funktionsbausteinen in einer dreistufigen Struktur „Erfassung Sensordaten“, „Verarbeitung [der Sensordaten]“ und „Ansteuerung Betätigungselemente“. Dieses

Entwurfsprinzip wird von der Matrixmethode des IFA umgesetzt (Abschnitt 5.7).

Es folgt die Tabelle J.1 des Anhangs, die eine beispielhafte Zusammenstellung von Aktivitäten und Dokumenten zur Anwendung des V-Modells zeigt. Diese Aktivitäten und Dokumente lassen sich ebenfalls in der IFA-Matrixmethode wiederfinden.

Ins Detail geht der Anhang mit der „Verifikation der Softwarespezifikation". Hier geht der Anhang offensichtlich davon aus, dass eine Softwarespezifikation textuell umgesetzt wurde. Textuelle Spezifikation führt leicht zu Lücken, Inkonsistenzen und fehlerhafter Interpretation – im Gegensatz zur Matrixmethode des IFA. Dort sind in deren Tabellendarstellungen auch formale Verifikationen möglich.

Im abschließenden Teil des Anhangs J werden Beispiele für Programmierregeln dargestellt. Solche konkreten Beispiele sind selten, zumal sich diese Regeln tatsächlich auf SRASW beziehen. Die Programmierregeln werden in der Matrixmethode ds IFA in der Tabelle „A3 Maßnahmen" (Beispiel: siehe Tabelle 9 in diesem Report) dokumentiert und verifiziert.

## 3.4 Relevante normative Inhalte der DIN EN ISO 13849-2:2013 zur SRASW

Nach der Entwicklung der SRASW gemäß DIN EN ISO 13849-1 folgt abschließend die Validierung der SRASW nach Teil 2 der Normenreihe [10]. Die Validierung umfasst mehrere Schritte während und am Ende des Entwicklungsprozesses. Sie wird als Nachweis der Eignung – bezogen auf den realen Einsatzzweck der Software – betrachtet. Durch Analysen und Tests wird also überprüft, ob die spezifizierten Sicherheitsanforderungen an die sicherheitsrelevanten Teile der Maschinensteuerung erreicht wurden. Die Matrixmethode des IFA enthält auch Elemente der Validierung und setzt damit die diesbezüglichen Anforderungen der DIN EN ISO 13849-2 um. Bis auf den Abschnitt 9.5 „Validierung der sicherheitsbezogenen Software" enthält Teil 2 der Norm keine weitere unmittelbare Bezugnahme auf SRASW. Die Aspekte der SRASW-Validierung werden in diesem Report in Kapitel 12 behandelt.

DIN EN ISO 13849-2 liefert als „Werkzeugkasten" auch eine umfassende Liste mit Maßnahmen gegen den systematischen Ausfall, die angewendet werden sollten, wie „Grundlegende Sicherheitsprinzipien" und „Bewährte Sicherheitsprinzipien". Für SRASW kann man folgende Sicherheitsprinzipien elektrischer Systeme anwenden (aus Anhang D der Norm):

- Vermeidung undefinierter Zustände: Undefinierte Zustände im Steuersystem sind zu vermeiden. Das Steuersystem ist konstruktiv so zu gestalten, dass während des üblichen Betriebs und unter allen erwarteten Betriebsbedingungen der Zustand des Steuersystems, z. B. die Ausgänge, vorherbestimmt werden kann.

- Zustandsausrichtung bei Ausfällen: Nach Möglichkeit sollten alle Einrichtungen/Schaltungen bei Ausfall in einen sicheren Zustand übergehen oder zu sicheren Bedingungen.

- Verringerung von Fehlermöglichkeiten: Trennung sicherheitsbezogener von anderen Funktionen.

- Gleichgewicht zwischen Komplexität/Vereinfachung: Ein Ausgleich sollte hergestellt werden zwischen der Komplexität der Einrichtungen, um eine bessere Steuerung zu erreichen, und der Vereinfachung der Einrichtungen, um ihre Zuverlässigkeit zu verbessern.

Im Teil 2 findet sich noch Tabelle D.21 „Fehler und Fehlerausschlüsse – Elektronische Bauteile – Programmierbare und/oder komplexe integrierte Schaltkreise" mit dem Eintrag, dass für „Fehler in allen Teilen der Funktion oder in einem Teil der Funktion einschließlich Software-Fehler" kein Fehlerausschluss erfolgen kann. Die Praxis bestätigt diese Festlegung.

Im Anhang E der DIN EN ISO 13849-2:2013 wird ein konkretes Beispiel für die Validierung einer programmierbaren Steuerung dargestellt, allerdings ohne die Validierung der Anwendungssoftware (siehe Anmerkung in Anhang E.1 der Norm). Diese beispielhafte Validierung der Anwendungssoftware wird in diesem Report im Abschnitt 12.3 dargestellt.

# 4 Risikobeurteilung und Sicherheitsfunktionen

Dieses Kapitel beschreibt, wie die Sicherheitsfunktionen und das Maß der fehlervermeidenden Anforderungen an SRASW ausgehend von der Risikobeurteilung abgeleitet werden.

## 4.1 Risikominderung durch Sicherheitsfunktionen

Die Norm DIN EN ISO 13849-1 wird dann angewendet, wenn als Teil der Risikominderung an einer Maschine eine steuerungsbasierte Schutzmaßnahme – die Sicherheitsfunktion – zu gestalten und zu bewerten ist. Neben der Steuerungshardware und der Embedded-Software übernimmt auch die SRASW einen Beitrag zur Risikominderung durch das realisierte Programm für die Sicherheitsfunktionen. Damit ist die Entwicklung der SRASW in den Rahmen der Risikobeurteilung gemäß ISO 12100 [13] und nationalen, europäischen Gesetzen und Richtlinien einzuordnen. Der Prozess der Risikobeurteilung wurde bereits im BGIA-Report 2/2008 [2], Kapitel 5 und Anhang A, beschrieben.

## 4.2 Festlegung von Sicherheitsfunktionen und deren Eigenschaften

Die korrekte und vollständige Definition der Sicherheitsfunktionen mit ihren Eigenschaften ist eine notwendige Voraussetzung für die Erstellung der SRASW und damit für die Anwendung der Matrixmethode des IFA. Die Beschreibung dieses Prozesses würde den Rahmen dieses Reports sprengen, daher sei auf das vom IFA veröffentlichte SISTEMA-Kochbuch 6 „Definition von Sicherheitsfunktionen“ [14] sowie den BGIA-Report 2/2008 [2], Kapitel 5 und Anhang A, verwiesen. Als Ergebnis liegt danach die Spezifikation der Sicherheitsfunktionen vor und wird als Eingangsdokument für die Matrixmethode verwendet.

## 4.3 Einfluss der Risikobewertung auf die Softwareentwicklung

Wie werden nun die normativen Anforderungen für SRASW ausgewählt? DIN EN ISO 13849-1 [1] fordert in Abschnitt 4.6.3 dazu: „*Bei SRASW für Bauteile mit einem $PL_r$ von a bis e müssen die folgenden Basismaßnahmen angewendet werden*“ und im weiteren Text „*Für SRASW in Komponenten mit $PL_r$ von c bis e werden die folgenden zusätzlichen Maßnahmen mit steigender Wirksamkeit … erforderlich oder empfohlen.*“

Das bedeutet also, dass sich die Anforderungen an dem $PL_r$ der realisierten Sicherheitsfunktion orientieren und nicht an dem PL (im Sinne von Bauteileeigenschaft) des verwendeten Steuerungsteils. Wenn also z. B. in Abbildung 3 eine Sicherheits-SPS (Steuerungsteil 2) mit einem typischen PL e spezifiziert ist, bedeutet dies nicht automatisch, dass auch die SRASW immer mit höchsten Anforderungen für $PL_r$ e entwickelt werden muss. Wird diese Sicherheits-SPS eingesetzt, um Sicherheitsfunktionen mit einem geringeren $PL_r$ c (mittleres Risiko; bei Sicherheitsfunktion 2) zu realisieren, dann genügen für die SRASW auf dieser SPS auch die Qualität und damit die Anforderungen für diesen $PL_r$ c.

Anmerkung: In Abbildung 3 werden durch die SRASW die beiden Sicherheitsfunktionen 1 und 2 realisiert. Der $PL_r$ beider Sicherheitsfunktionen ist unterschiedlich (a und c). Da der $PL_r$ von Sicherheitsfunktion 2 der höhere ist, bestimmt er die Anforderungen an die SRASW.

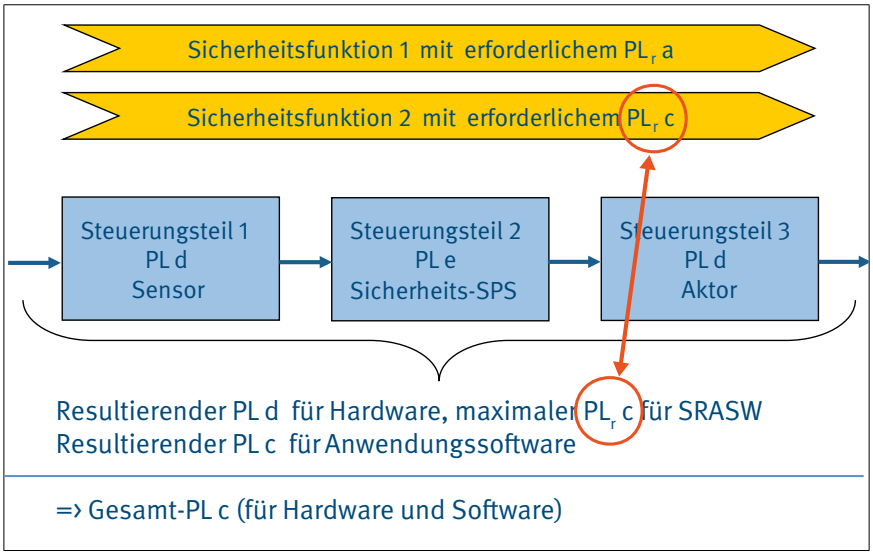



Abbildung 3:
Beispiel für Herleitung der Anforderungen an SRASW

## 4.4 Einfluss der Softwarestruktur auf die Softwareentwicklung

Meistens werden auf einer programmierbaren Steuerung mehrere Sicherheitsfunktionen realisiert, die durchaus unterschiedliche $PL_r$ haben können. Dann kann theoretisch auch die SRASW für jede Sicherheitsfunktion entsprechend ihrem $PL_r$ qualitativ unterschiedlich entwickelt werden. Diese Trennung ist aber für typische SRASW, für die die Matrixmethode des IFA eingesetzt werden kann, kaum praktikabel. DIN EN ISO 13849-1 fordert dazu in Abschnitt 4.6.3:

„*Wenn ein Teil der SRASW innerhalb eines Bauteils irgendeinen Einfluss (z. B. bei Modifikation) auf verschiedene Funktionen mit unterschiedlichen PL hat, dann müssen die Anforderungen des zugehörigen höchsten PL angewendet werden.*“

Wie kann im speziellen Fall nachgewiesen werden, dass Teile der SRASW sich nicht gegenseitig beeinflussen und damit unterschiedliche Anforderungen innerhalb der SRASW doch anwendbar sind? Es muss dann Folgendes dargelegt und dokumentiert werden:

- Es ist eine Unabhängigkeit der einzelnen Sicherheitsfunktionen sowohl im räumlichen und zeitlichen Bereich gegeben oder
- jede Verletzung dieser Unabhängigkeit wird beherrscht.

Die Rechtfertigung für diese Unabhängigkeit muss dokumentiert werden. Typische Merkmale zur Unabhängigkeit von Softwaremodulen und entsprechende Techniken werden im Abschnitt 5.9 beschrieben.

Die Matrixmethode des IFA ist zunächst unabhängig vom $PL_r$ einzusetzen. Die Auswahl und Anwendung von fehlervermeidenden Techniken und Maßnahmen richtet sich vereinfachend nach dem höchsten $PL_r$ der betrachteten Sicherheitsfunktionen und wird von dem Tool SOFTEMA (Kapitel 14) unterstützt.

## 4.5 Einfluss der Hardwarezuverlässigkeit auf die Softwareentwicklung

In Abbildung 3 wird noch deutlich, dass der PL für die gesamte Kombination der drei Steuerungsteile immer aus zwei Aspekten folgt:

- der Zuverlässigkeit der Hardware: Sie ergibt sich in diesem Beispiel zu einem PL aus der Kombination der drei Steuerungsteile mit PL d/e/d. Daraus resultiert in diesem Beispiel ein Gesamt-PL d für die Hardware.
- der Zuverlässigkeit der Software: Angenommen, die SRASW der Steuerungsteile 2 und 3 würden entsprechend der Anforderungen für $PL_r$ c entwickelt (würden also nicht mehr $PL_r$ d entsprechen), dann ergäbe sich für die Sicherheitsfunktion folgende Situation: $PL_{Hardware}$ d mit $PL_{Software}$ c, das ergibt aufgrund der „schlechteren“ Software eine Abstufung auf Gesamt-PL c.

Ergibt es sich in diesem Beispiel nachträglich, dass eine weitere Sicherheitsfunktion mit einem höheren $PL_r$ d zu realisieren ist, dann hat dies unter Umständen Konsequenzen für die bereits realisierten Anwendungsprogramme. Aufgrund der oben zitierten Anforderung müssen bei fehlender Trennung der Software-Sicherheitsfunktionen untereinander diese eventuell alle nachträglich „ertüchtigt“ werden. Daher sollte das Qualitätsziel, also der vorgegebene $PL_r$ für die Softwareentwicklung, nicht zu knapp vorgegeben werden. Im Beispiel der Abbildung 3 wäre es sinnvoll, entsprechend der Hardwarequalität auch die Softwarequalität mit $PL_r$ d zu erreichen. So könnte später ohne größeren Aufwand auch eine Sicherheitsfunktion mit $PL_r$ d ergänzt werden.

# 5 Fehlervermeidende Maßnahmen

Treten in einer Steuerung während des Betriebes Ausfälle auf, dann entweder durch zufällige Fehler aufgrund von Alterung oder physikalischen Phänomenen oder aufgrund von systematischen Fehlern, die schon vor der Inbetriebnahme entstanden sind, z. B. Spezifikations-, Implementierungs- oder Fertigungsfehler. Bei der Anwendungssoftware treten nur systematische Fehler auf. Wenn diese systematischen Fehler sich im Betrieb negativ auswirken, dann können eventuell sogenannte fehlerbeherrschende Maßnahmen in der eingebetteten Software (z. B. Zykluszeitüberwachung mit einem Hardware-Watchdog) einen sicheren Zustand herbeiführen. Besser ist es, die systematischen Fehler gar nicht erst entstehen zu lassen. Dazu dienen während der Softwareentwicklung die fehlervermeidenden Maßnahmen. In diesem Kapitel werden die wichtigsten Maßnahmen und Techniken aus DIN EN ISO 13849-1 vorgestellt und kommentiert. Im folgenden Abschnitt wird zunächst der typische Ablauf einer gemeinsamen Projektierung von Steuerungshardware und -software dargestellt.

## 5.1 Typischer Projektablauf

Der Prozess der Risikominderung muss den gesamten Lebenszyklus einer Maschine berücksichtigen. Der Entwicklungsablauf für sicherheitsbezogene Steuerungen ist darin enthalten. Für sicherheitsbezogene Software ist dieser Ablauf durch das sogenannte „V-Modell“ (Abschnitte 5.2 bis 5.5 dieses Reports) sogar sehr konkret vorgegeben. Die Abschnitte ab 5.6 stellen weitere grundlegende Entwicklungsaspekte im Rahmen des V-Modells für SRASW vor.

Allerdings „benötigt“ Software auch immer eine Steuerungshardware, um komplette Sicherheitsfunktionen zu realisieren. Wie diese gemeinsame, sich bedingende Entwicklung von Hardware und Anwendungssoftware ablaufen sollte, ist in der Norm nicht explizit angegeben. Dieser Abschnitt beschreibt plakativ einen möglichen Projektablauf bei der Realisierung von Sicherheitsfunktionen. Abbildung 4 zeigt eine Übersicht der typischen Projektschritte.

Da der Fokus auf der Realisierung von Sicherheitsfunktionen liegt, startet der Ablaufplan bei der Spezifikation der Sicherheitsfunktionen. Als Grundlage für die Spezifikation der Sicherheitsfunktionen dienen die Konstruktionsdaten der Anlage und die Spezifikation der Sicherheitsanforderungen für die ganze Maschine (oft als Sicherheitskonzept bezeichnet). Dazu gehört auch die Anlagenskizze (Dokument A2.1). Alle hier benannten Dokumententypen Ax.x, Bx.x, usw. sind im Abschnitt 5.5 genauer beschrieben.

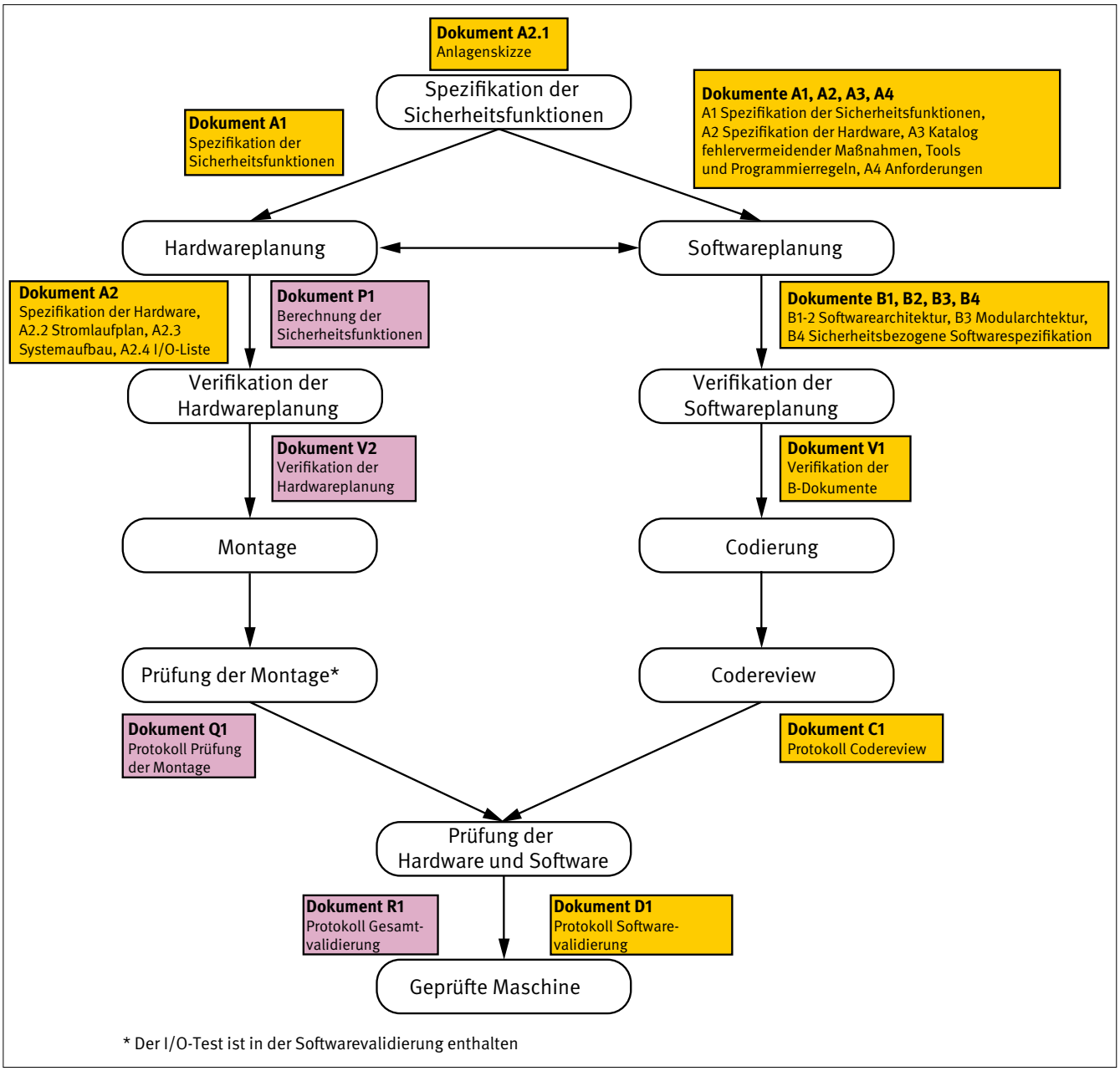
Der Projektablauf gliedert sich danach in einen hardware- und in einen softwareorientierten Zweig. Die für die Anwendungsprogrammierung benötigten Dokumente A bis D und V1 sind gelb hinterlegt. Die während der Hardwareplanung zusätzlich anfallenden Dokumente, die in diesem Report nicht näher durch Beispiele belegt werden, sind pinkfarben unterlegt.

Die Entwicklung sicherheitsbezogener Steuerungsteile beginnt mit der Spezifikation der Sicherheitsfunktionen (Dokument A1). Darauf basierend werden in der Hardwareplanung der Stromlaufplan (Dokument A2.2), der Systemaufbau (Dokument A2.3) und die I/O-Liste (Dokument A2.4) erstellt. Weiterhin erfolgt die Berechnung der Performance Levels der Sicherheitsfunktionen (Dokument P1), z. B. mit der IFA-Software SISTEMA [15], siehe auch BGIA-Report 2/2008 [2]. Anschließend wird die Hardwareplanung gegen die Spezifikation der Sicherheitsfunktionen verifiziert (Dokument V2). Die verifizierten Hardwareplanungsunterlagen dienen als Grundlage für die Montage. Die Montage wird geprüft und das Ergebnis dokumentiert (Dokument Q1).

Als Grundlage für die Softwareplanung dienen ebenfalls die Spezifikation der Sicherheitsfunktionen A1 und die Dokumente A2.1 bis A2.4 der Hardwareplanung. Zusätzlich sind der Katalog fehlervermeidender Maßnahmen, die Tools und Programmierregeln (Dokument A3) sowie normative Anforderungen (Dokument A4) als Planungsinformationen verfügbar. Während der Softwareplanung entstehen die Dokumente Architektur Sicherheitsprogramm (Dokument B1), Architektur Standardprogramm (Dokument B2), Modularchitektur (Dokument B3) und die sicherheitsbezogene Softwarespezifikation mit dem Validierungsplan (Dokument B4). Diese Dokumente müssen anschließend gegen die Spezifikation der Sicherheitsfunktionen verifiziert werden (Dokument V1). Anschließend folgt die Programmierung in der Phase Codierung und das Überprüfen des Programmcodes beim Codereview (Dokument C1). Einige der genannten Dokumente sind optional.

Nach der Prüfung der Montage und des Codereviews kann die Prüfung der Hard- und Software erfolgen (Dokumente D1 und R1). Diese wird in Prüfungsdokumenten festgehalten. Danach erfolgt die Gesamtvalidierung, die in Abbildung 4 nicht separat dargestellt ist. Die einzelnen Verifikationen und Prüfungen sind vorzugsweise jeweils nach dem Vier-Augen-Prinzip von einer weiteren Person, die über einschlägige Qualifikation verfügt, durchzuführen (Abschnitt 5.15).

Abbildung 4:
Typischer Ablaufplan für Projektierung mit programmierbaren Steuerungen

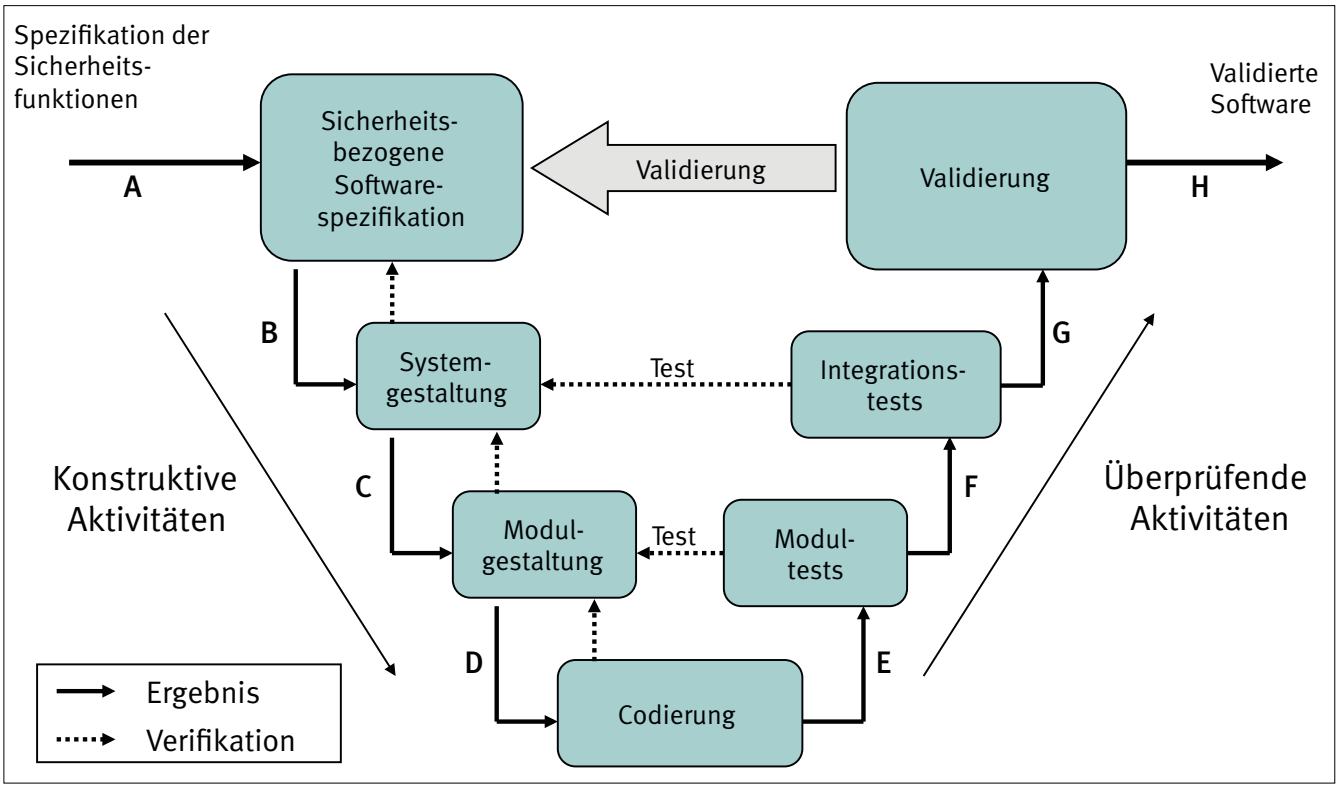



## 5.2 Entwicklungsmodell V-Modell

Das V-Modell ist im Bereich der Funktionalen Sicherheit eine normenübergreifende Darstellung eines Entwicklungsprozesses für sicherheitsbezogene Software. Es wurde von *Boehm* bereits 1979 [16] in die Softwaretechnik eingeführt. Die Sicherheits-Grundnormenreihe IEC 61508 hat dieses Modell aufgegriffen und in einer sehr detaillierten Form dargestellt. Dies war notwendig, da der Anwendungsbereich dieser Grundnorm zwangsläufig sehr weit ist: Er reicht von einfachen Steuerungsgeräten für eine handgeführte Maschine bis hin zu hochkomplexen Leitsystemen für prozesstechnische Anlagen. Letztere erfordern diese Detaillierung im Entwicklungsmodell.

Wenn allerdings im Maschinensektor dedizierte Steuerungsgeräte mit integrierten Entwicklungsumgebungen, wie z. B. Sicherheits-SPS, verwendet werden, dann kann das V-Modell ausgehend von seiner komplexen Form in der IEC 61508 vereinfacht werden. Die Phase „Softwarearchitektur“ kann in der Anwendungsprogrammierung entfallen, da durch das Betriebssystem und das Entwicklungswerkzeug der Steuerung die Softwarearchitektur vorgegeben ist. Dies könnte z. B. eine Architektur gemäß der relevanten SPS-Norm DIN EN 61131-3 [12] sein, die vom Steuerungshersteller realisiert und von Prüfstellen zertifiziert wurde. Dementsprechend kann im aufsteigenden, testenden Teil des V-Modells auf die Phase „Integrationstests“ in Bezug auf die Hardware verzichtet werden. Dem Anwendungsprogrammierenden bleibt, die Integration der Softwaremodule untereinander und in die vorgegebene Softwarearchitektur zu testen. Dies ist auch der Hintergrund für das „vereinfachte V-Modell“ in DIN EN ISO 13849-1, Abschnitt 4.6.1 (Abbildung 5).

Abbildung 5:
Das vereinfachte V-Modell aus DIN EN ISO 13849-1, Abschnitt 4.6.1




## 5.3 Beschreibung des V-Modells

Dieser Abschnitt soll zusammengefasst die Anforderungen aus DIN EN ISO 13849-1, Abschnitt 4.6 darstellen. Eine zentrale Forderung der Norm besteht darin, dass die Entwicklung der sicherheitsbezogenen Anwendungssoftware nach dem vereinfachten V-Modell erfolgen soll (Abbildung 5).

Die Anwendung des V-Modells verfolgt zwei Ziele:

- Vermeidung von Softwarefehlern (systematischen Fehlern) und
- Entwicklung von lesbarer, verständlicher, testbarer und wartbarer Software.

Das V-Modell lässt sich folgendermaßen charakterisieren:

- Das V-Modell besteht aus konstruktiven (Abbildung 5, links, inkl. Codierung) und überprüfenden Aktivitäten, auch Phasen genannt (Abbildung 5, rechts).
- Das Ergebnis jeder konstruktiven Phase muss gegenüber den Vorgaben der Vorgängerphase verifiziert werden. Die Ergebnisse sind mit den Buchstaben A bis H in Abbildung 5 gekennzeichnet.
- Jeder konstruktiven Phase steht eine überprüfende Phase gegenüber. Der Testplan dazu ist bereits in der konstruktiven Phase entwickelt worden.

Zur Minimierung von Fehlern in der Softwareentwicklung dient eine saubere Dokumentation der erstellten Software während aller Phasen des V-Modells. Die Spezifikation und der Softwareentwurf müssen dokumentiert werden, damit später das eigentliche Programm gegen diesen Entwurf verifiziert werden kann. Diese Spezifikation sollte allen an der Softwareentwicklung Beteiligten zugänglich sein. Zur Spezifikation gehört eine Definition der Sicherheitsfunktionen, in denen der erforderliche Performance Level $PL_r$ und die zugehörige Betriebsart festgelegt werden.

Die Programmierung soll modular und strukturiert erfolgen. Dabei bietet es sich an, wo immer möglich auf validierte Funktionsbausteine der Hersteller zurückzugreifen. Bei der Programmierung sollte viel Wert auf lesbaren und verständlichen Code gelegt werden, dazu sollten unbedingt symbolische Variablennamen verwendet und Programmierrichtlinien eingehalten werden (siehe auch Abschnitt 5.7). Verständlicher Code macht einen Test des Programms einfacher und vermeidet zusätzliche Softwarefehler bei späteren Modifikationen.

Einige Anforderungen aus der Norm werden von den zertifizierten SPS-Programmierumgebungen direkt erfüllt:

- Benutzung von LVL-Programmiersprachen,
- Trennung von Sicherheitssoftware und rein funktionaler Software,
- Bereitstellung von validierten Funktionsbausteinen.

Bei Änderungen in der Software müssen geeignete Entwicklungsmaßnahmen stattfinden. Dies umfasst die Dokumentation aller Änderungen und eine Einflussanalyse. Anhand der Einflussanalyse kann der Aufwand für die erneut notwendigen

Aktivitäten nach dem V-Modell abgeschätzt werden. Alle Änderungen müssen in einer Änderungshistorie dokumentiert werden.

Die Verifikation umfasst die Analysen und Funktionstests für SRP/CS bzw. deren Teilaspekte, die feststellen, ob die erzielten Resultate einer Entwicklungsphase den Vorgaben für diese Phase entsprechen. Im Rahmen des Codereviews wird z. B. überprüft, ob die Anwendungssoftware den Vorgaben der fehlervermeidenden Maßnahmen entspricht.

Bei der Softwarevalidierung geht es um den Nachweis, dass die Anwendungssoftware die Spezifikationen erfüllt. Die Softwarevalidierung setzt sich aus Analysen und Funktionstests zusammen (siehe Kapitel 12). Die daran anschließende Gesamtvalidierung der Sicherheitsfunktionen (nicht im V-Modell dargestellt) geht u. a. zusätzlich der Frage nach, ob auch die richtigen Sicherheitsfunktionen für die Maschine festgelegt worden sind („Ist das Richtige gebaut worden?“).

Alle diese Anforderungen werden von Programmierenden, die üblicherweise nicht sicherheitsrelevante Anwendungssoftware erstellen, zunächst als mühsam empfunden, geben ihnen aber andererseits die Bestätigung, die Software hinreichend gut entwickelt zu haben.

## 5.4 Vereinfachung des V-Modells für typische SRASW

Für die in der Norm verlangte Entwicklung der Anwendungssoftware nach dem V-Modell sind noch Vereinfachungen möglich. Abbildung 6 deutet gestrichelt die Bereiche an, die zusammengefasst werden können. Diese Vereinfachungen gelten für unten angegebene Rahmenbedingungen. Die sicherheitsgerichtete Anwendungssoftware für Maschinen ist meistens nach einer typischen Struktur aufgebaut. Sie gliedert sich in eine Vorverarbeitungsebene, eine Ansteuerlogik und eine Ansteuerungsebene (siehe Seite 33). Die Vorverarbeitungs- und Ansteuerungsebenen nutzen oft zertifizierte Funktionsbausteine, die die Steuerungshersteller in Bibliotheken zur Verfügung stellen. Es muss nur noch die Ansteuerlogik der Funktionsbausteine der Ansteuerungsebene spezifiziert und näher getestet werden. Die Verschaltung aller Funktionsbausteine mit den sonstigen Peripheriesignalen ist klar und von einfacher Natur. Durch diese vorgegebene Struktur ist die Systemgestaltung überwiegend vorweggenommen. Dies begründet die Zusammenfassung der ersten und letzten beiden Phasen, da die „Systemgestaltung“ und ihre korrespondierende Testphase „Integrationstest“ in der Praxis wegfallen.

Weiterhin ist es sinnvoll, die ggf. benötigte Entwicklung von eigenen Funktionsbausteinen in ein separates V-Modell auszulagern. Diese selbst entwickelten und validierten Funktionsbausteine werden später bei der Programmierung von Sicherheitsfunktionen gleichwertig zu den von den Herstellerfirmen bereitgestellten Bibliotheksbausteinen verwendet.

Nimmt man diese Zusammenfassungen vor, so ergeben sich zwei kleine V-Modelle: eines für die Softwareentwicklung von kompletten Sicherheitsfunktionen und eines für die Softwareentwicklung einzelner Softwaremodule (Funktionsbausteine) von SRASW (Abbildung 7). Letzteres kann in einer Anwendung oft notwendig sein, wenn eine erforderliche Funktionalität nicht durch Hersteller-Bibliotheksbausteine abgedeckt werden kann. Es ist dann sinnvoll, wenn diese in einem Funktionsbaustein gekapselt in einer Bibliothek zur Verfügung steht, z. B. um mehrfach wiederverwendet werden zu können.

Abbildung 6:
Das V-Modell mit möglichen Zusammenfassungen (gestrichelt)

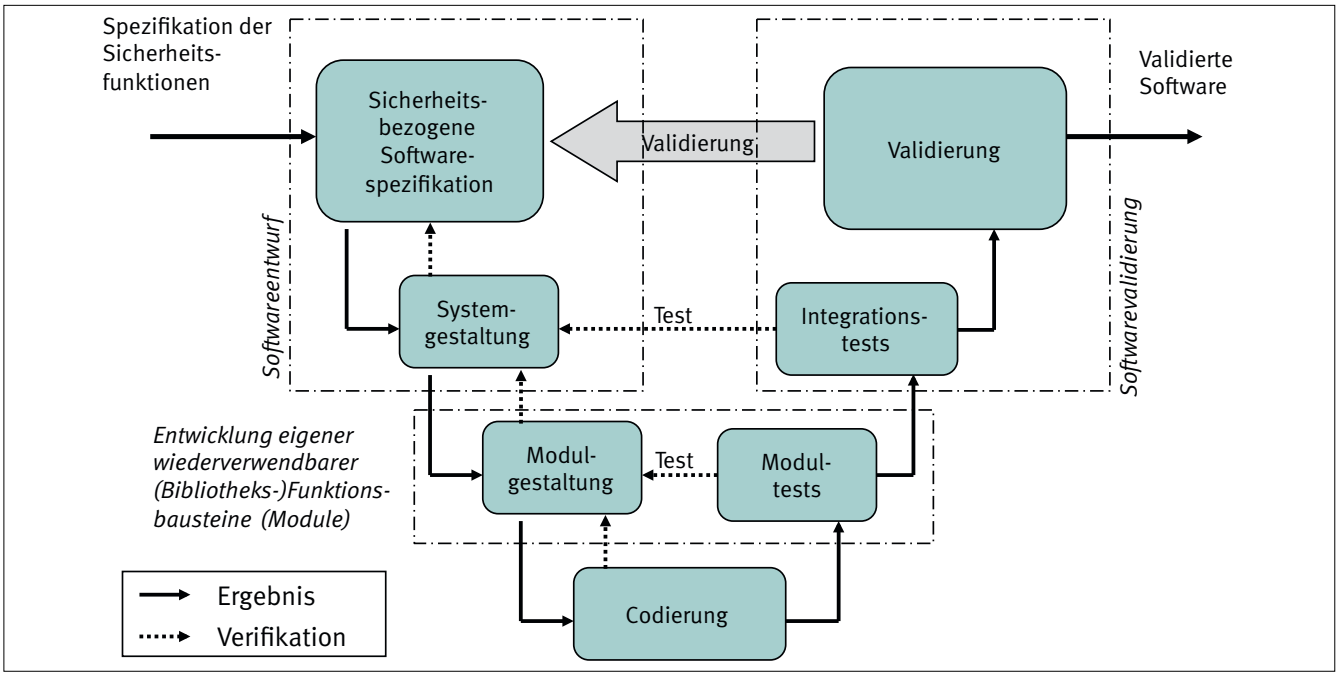
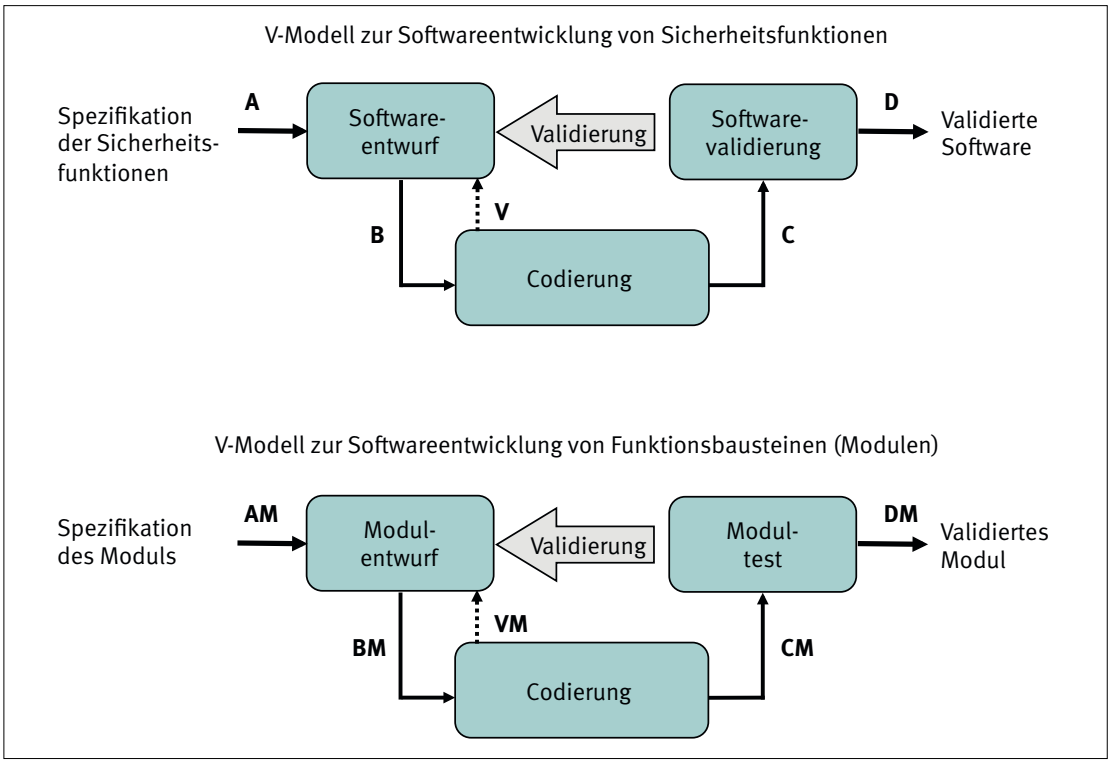


Abbildung 7:
Zusammengefasste V-Modelle für Sicherheitsfunktionen (oben) und von Modulen (unten)




## 5.5 Dokumententypen für das vereinfachte V-Modell

Für die Softwareentwicklung nach einem V-Modell ist es praktikabel, Dokumententypen für die Ein- und Ausgänge der Phasen zu definieren. Für die Nachverfolgbarkeit dieser Dokumententypen dienen bei der Matrixmethode des IFA die Buchstaben A bis D und V an den Ein- und Ausgängen der Phasen in Abbildung 7. Beim V-Modell zur Modulentwicklung ist jeweils der Buchstabe „M“ hinzugefügt, um Verwechslungen zu vermeiden. Die Tabellen 1 und 2 definieren die Dokumententypen für die beiden V-Modelle.

Tabelle 1:
Dokumententypen für das V-Modell zur Softwareentwicklung von Sicherheitsfunktionen

| Kürzel | Dokument | Anmerkung |
|---|---|---|
| A1 | Spezifikation der Sicherheitsfunktionen | vorhanden |
| A2.1 | Spezifikation der Hardware, Anlagenskizze | optional |
| A2.2 | Spezifikation der Hardware, Stromlaufplan | vorhanden |
| A2.3 | Spezifikation der Hardware, Systemaufbau | vorhanden |
| A2.4 | Spezifikation der Hardware, I/O-Liste | vorhanden |
| A3 | Katalog fehlervermeidender Maßnahmen, Tools und Programmierregeln | wiederverwendbar |
| A4 | Anforderungen | wiederverwendbar |
| B1 | Architektur Sicherheitsprogramm | entfällt bei einfachen Anwendungen |
| B2 | Architektur Standardprogramm | entfällt bei einfachen Anwendungen, optional |
| B3 | Modularchitektur | |
| B4 | Sicherheitsbezogene Softwarespezifikation und Validierungsplan | |
| B5 | Programmskizze | optional |
| | | |
| V1 | Protokoll Verifikation | |
| | | |
| C1 | Protokoll Codereview | |
| | | |
| D1 | Protokoll Softwarevalidierung | |

Tabelle 2:
Dokumententypen für das V-Modell zur Softwareentwicklung von Modulen (Funktionsbausteinen)

| Kürzel | Dokument | Anmerkung |
|---|---|---|
| AM1 | Schnittstellendefinition und Funktionsbeschreibung | |
| AM2 | Katalog fehlervermeidender Maßnahmen, Tools und Programmierregeln | wiederverwendbar |
| AM3 | Anforderungen | wiederverwendbar |
| BM1 | Modulspezifikation und Testplan | |
| BM2 | Programmskizze | optional |
| | | |
| VM1 | Protokoll Verifikation | |
| | | |
| CM1 | Protokoll Codereview | |
| | | |
| DM1 | Protokoll Modultest | |

In Tabelle 1 sind diejenigen Dokumente mit „vorhanden“ gekennzeichnet, die bei einem Projekt in jedem Fall, z. B. für die Hardwareplanung, erstellt werden müssen. „Wiederverwendbar“ bedeutet, dass dieses Dokument auch in anderen Projekten angewendet werden kann. Es muss also nicht für jedes Projekt neu erstellt werden. Die optionalen Dokumente benötigt dieser Report in den Beispielen, um die Vorgehensweise verständlich zu erklären, sie sind aber im Projekt meist entbehrlich.

Im Folgenden werden die Dokumente kurz beschrieben:

- A1 – Spezifikation der Sicherheitsfunktionen: tabellarische Auflistung der Sicherheitsfunktionen mit wichtigen Eigenschaften wie z. B. erforderlicher Performance Level, Reaktionszeiten, Betriebsarten, Priorität usw.,
- A2.1 – Anlagenskizze: Grobübersicht der Anlage, kann auch die Konstruktionsdaten enthalten,
- A2.2 – Stromlaufplan: elektrische Verschaltung insbesondere sicherheitstechnischer Komponenten,
- A2.3 – Systemaufbau: Übersicht der Sicherheitskomponenten und deren Verbindung (Topologie mit Netzwerk),
- A2.4 – I/O-Liste: Liste aller sicherheitsrelevanten und ggf. anderer relevanter Ein- und Ausgänge mit Adressen und Variablennamen. Die I/O-Liste enthält auch Prüffelder für das Codereview und die Validierung der Software.
- A3 – Katalog fehlervermeidender Maßnahmen, Tools und Programmierregeln zur Fehlervermeidung,
- A4 – Anforderungen: Auflistung und projektspezifische Kommentierung der normativen Anforderungen gemäß DIN EN ISO 13849-1, Abschnitt 4.6.3,
- B1 – Architektur Sicherheitsprogramm: Übersicht über den Aufbau des Sicherheitsprogramms (Aufrufhierarchie von Bausteinen),
- B2 – Architektur Standardprogramm: Übersicht über den Aufbau des Standardprogramms (Aufrufhierarchie von Bausteinen),
- B3 – Modularchitektur: Übersicht der benutzten Module (Funktionsbausteine) mit den verschalteten Ein- und Ausgängen und den Logiksignalen für die Ansteuerlogik,
- B4 – Sicherheitsbezogene Softwarespezifikation und Validierungsplan: Matrixbasierte Spezifikation der Software und Testplan für die Verifikation und Validierung der Software,
- B5 – Programmskizze: Darstellung der Software z. B. im Funktionsbausteindiagramm; entspricht in der Regel dem späteren Programmlisting,
- V1 – Protokoll Verifikation: kein eigenes Dokument, sondern Prüffelder in den Dokumenten B3 und B4, da diese gegen die Spezifikation der Sicherheitsfunktionen (A1) verifiziert werden müssen,
- C1 – Protokoll Codereview: Der Softwarecode muss durch Review auf Fehler untersucht werden. Im Protokoll Codereview sind die Verifikationsschritte aufgelistet und Fehler zu dokumentieren.
- D1 – Protokoll Softwarevalidierung: Protokoll für die Validierung der Software.

Für die Softwareentwicklung von Modulen:

- AM1 – Schnittstellendefinition und Funktionsbeschreibung: Beschreibung der Ein- und Ausgangsvariablen eines Moduls (Funktionsbaustein) und dessen Funktion,
- AM2 – Katalog fehlervermeidender Maßnahmen, Tools und Programmierregeln zur Fehlervermeidung,
- AM3 – Anforderungen: Auflistung und projektspezifische Kommentierung der normativen Anforderungen der DIN EN ISO 13849-1, Abschnitt 4.6.3.,

- BM1 – Modulspezifikation und Testplan: entspricht B4. Matrixbasierte Spezifikation des Funktionsbausteins mit Testplan (nur bei einfachen Funktionsbausteinen möglich),
- BM2 – Programmskizze: Darstellung der Software, z. B. im Funktionsbausteindiagramm; entspricht in der Regel dem späteren Programmlisting,
- VM1 – Protokoll Verifikation: kein eigenes Dokument, sondern Prüffelder in dem Dokument BM1, die die Verifikation der Modulspezifikation BM1 gegenüber Funktionsbeschreibung AM1 dokumentieren,
- CM1 – Protokoll Codereview: Der Softwarecode muss durch Review auf Fehler untersucht werden. Im Protokoll Codereview sind die Verifikationsschritte aufgelistet und Fehler zu dokumentieren.
- DM1 – Protokoll Modultest: entspricht D1. Protokoll der Validierung eines Funktionsbausteins.

## 5.6 Spezifikation der Sicherheitsanforderungen und Sicherheitsfunktionen

Die Gestaltung und Integration sicherheitsbezogener Steuerungsteile muss daran orientiert sein, möglichst fehlerfreie, den Sicherheitsanforderungen entsprechende Produkte zu entwickeln und diese auch – wie vorgesehen – einzusetzen.

Mit der Spezifikation der Sicherheitsanforderungen für eine Maschine beginnt der Lebenszyklus der sicherheitsbezogenen Steuerungsteile (Abbildung 4) und damit der SRASW. Im BGIA-Report 2/2008 [2] ist im Abschnitt 6.1.1, Kasten 6.1 ein allgemeines Gliederungsschema für eine Spezifikation der Sicherheitsanforderungen dargestellt.

Die Spezifikation der Sicherheitsfunktionen ist ein Teil dieser Sicherheitsanforderungen und das erste wichtige Dokument für die SRASW-Entwicklung (siehe auch SISTEMA-Kochbuch 6 „Definition von Sicherheitsfunktionen“ [14]). DIN EN ISO 13849-1 listet in Kapitel 5 neben speziellen Aspekten verschiedener Sicherheitsfunktionen auch allgemeine Aspekte auf, die in einer solchen Spezifikation mindestens enthalten sein müssen. Das Dokument „A1 – Spezifikation der Sicherheitsfunktionen“ enthält bereits diese Angaben und kann um zusätzliche projektspezifische Angaben erweitert werden. Bei der Nutzung des IFA Tools SOFTEMA (Kapitel 14) wird jede Änderung und Erweiterung dieser Spezifikation A1 an die dann folgenden SRASW-Entwicklungsdokumente weitergeleitet.

Mit einer solchen Spezifikation werden für alle Beteiligten am Anfang des Entwicklungsprozesses (Abbildung 4) die Rahmenbedingungen festgelegt – es handelt sich um ein sogenanntes Lastenheft und keinesfalls um eine erst nach der Entwicklung angefertigte Produktbeschreibung.

## 5.7 Programmierrichtlinien

Während der Codierung ist es besonders wichtig, lesbaren und verständlichen Code zu schreiben, damit dieser später leicht getestet und fehlerfrei modifiziert werden kann. Dies gewährleisten geeignete und verbindliche Programmierrichtlinien, die bei der Codierung beachtet werden. Die Richtlinien sollten bestehende und akzeptierte Regeln einer anerkannten Organisation sein [8; 15; 17]. In DIN EN ISO 13849-1, Anhang J.4 sind ebenfalls Beispiele für Regeln gelistet. Alternativ kann ein Unternehmen anhand dieser Vorlagen selbst geeignete Programmierregeln zusammenstellen. Im Jahr 2013 startete eine weitere Initiative der PLCopen[1] im Promotional Committee 2 mit dem Arbeitskreis „Software Construction Guidelines“. Diese Guidelines sind bereits verfügbar.

Programmierrichtlinien regeln die Benutzung kritischer Sprachkonstrukte, den Umfang und die Schnittstelle von Softwarefunktionen, die Formatierung und Kommentierung des Codes, symbolische Namen von Funktionen und Variablen usw. In der Matrixmethode des IFA ist eine eigene Tabelle (Dokument A3) für die Darstellung und Verifizierung der Regeln vorgesehen (Beispiele in Abschnitt 6.5).

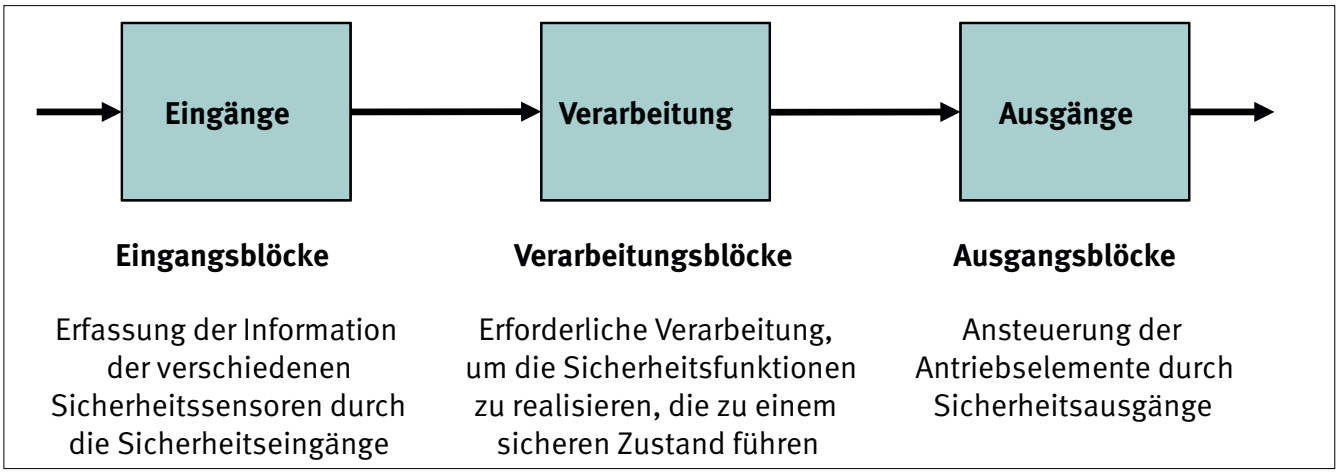
Diese allgemein gültigen Regeln werden in einem konkreten Projekt noch durch herstellerspezifische Programmierregeln und Konfigurationsmaßnahmen ergänzt, die die Besonderheiten der verwendeten Steuerungsfamilie und Softwarearchitektur behandeln (Beispiele in Abschnitt 6.5).

## 5.8 Modulare und strukturierte Programmierung

Modularität und Strukturierung sind grundlegende Prinzipien zur Fehlervermeidung und -beherrschung bei der Softwareentwicklung für sicherheitskritische Systeme. Daher spezifiziert dies die Norm DIN EN ISO 13849-1 schon als Basismaßnahme und liefert dazu auch ein Software-Architekturmodell in drei Stufen: Eingänge -› Verarbeitung -› Ausgänge (Abbildung 8). Diese drei Stufen sollten wiederum überwiegend durch Funktionsbausteine realisiert werden.

[1] PLCopen ist eine weltweit agierende, hersteller- und produktunabhängige Organisation für die Automatisierungsindustrie (www.plcopen.org)

Abbildung 8:
Allgemeines Architekturmodell für Software (nach DIN EN ISO 13849-1:2007, Bild 7)




Was sind nun die Definitionen dieser beiden Prinzipien? Dazu sei auf die Serie der Grund-Sicherheitsnorm DIN EN 61508 [3] verwiesen, die in Teil 7 den Überblick über Verfahren und Maßnahmen gibt und diese definiert. Im Folgenden daraus die Zitate:

„*Ziel* [der strukturierten Programmierung]*: Entwurf und Implementierung des Programms derart, dass die Analyse ohne Ausführung durchführbar wird. Das Programm darf nur so wenig wie möglich statisch nicht testbares Verhalten beinhalten.*

*Beschreibung: Die folgenden Prinzipien sollten zur Verringerung struktureller Komplexität angewendet werden:*

- *Unterteilung des Programms in angemessen kleine Softwaremodule, um sicherzustellen, dass diese so weit wie möglich entkoppelt und alle Wechselwirkungen deutlich sind;*
- *Entwurf des Steuerflusses der Softwaremodule unter Verwendung strukturierter Konstrukte, diese sind Abfolge (Sequenz), Wiederholung (Iteration) und Auswahl (Selektion);*
- *möglichst wenig Pfade durch ein Softwaremodul und einfache Beziehungen zwischen den Eingabe- und Ausgabeparametern;*
- *Vermeidung komplizierter Verzweigungen und insbesondere unbedingter Sprünge (GOTO) in Hochsprachen;*
- *wenn möglich, sind Abbruch- und Aussprungbedingungen von Schleifen an Eingabeparameter zu knüpfen;*
- *Vermeidung komplizierter Berechnungen als Grundlage von Verzweigungen und Schleifenbedingungen.*
- *Diejenigen Eigenschaften einer Programmiersprache, die das oben genannte Vorgehen fördern, sollten gegenüber anderen, die (angeblich) leistungsfähiger sind, bevorzugt verwendet werden, außer wenn Effizienz absolute Priorität hat (zum Beispiel bei einigen sicherheitskritischen Systemen).*“ (Zitat DIN EN 61508-7:2011 [3])

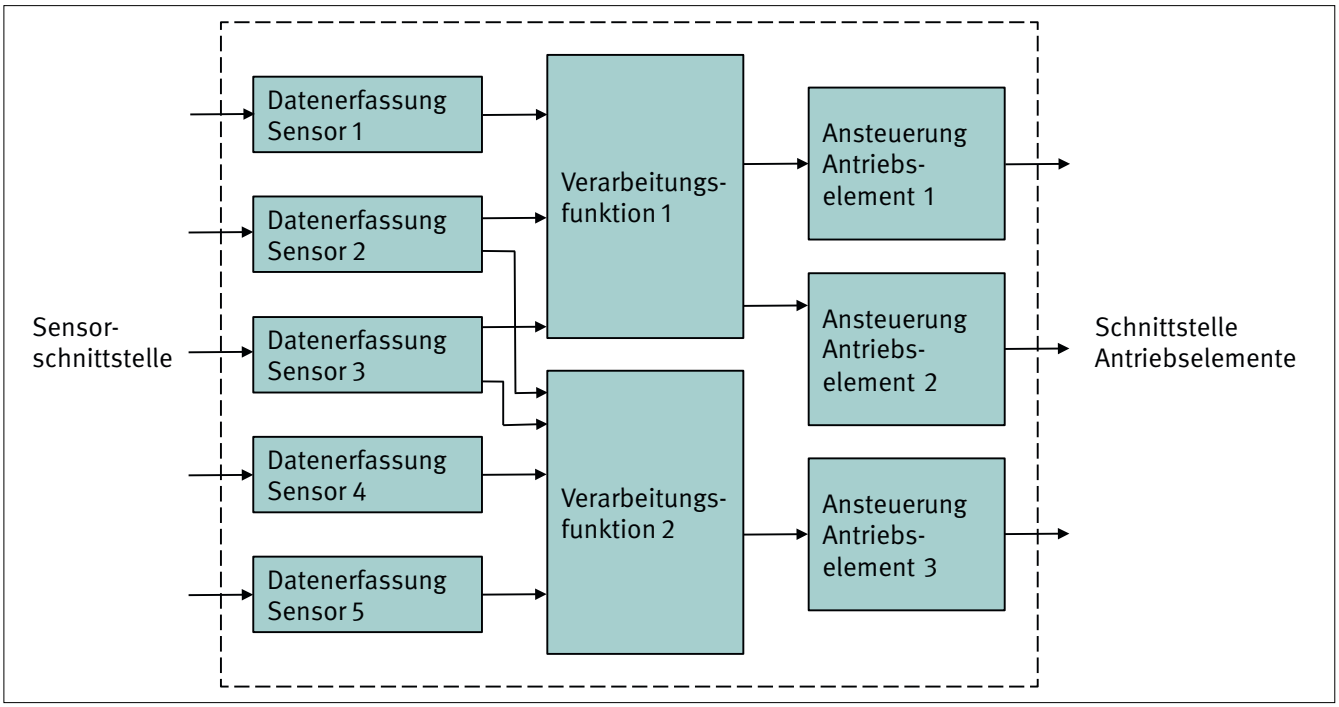
Das Prinzip der Modularisierung ist übergeordnet und bezieht sich mehr auf die Schnittstellen zwischen den Softwaremodulen:

„*Ziel* [der Modularisierung]*: Reduzierung der Komplexität und Vermeidung von Ausfällen bezogen auf Schnittstellen zwischen Teilsystemen.*

*Beschreibung: Auf allen Ebenen des Entwurfs wird jedes Teilsystem klar definiert und ist in der Größe beschränkt (nur wenige Funktionen). Die Schnittstellen zwischen Teilsystemen werden so einfach wie möglich gehalten und Querverbindungen (d. h. gemeinsame Daten, Informationsaustausch) werden gering gehalten. Die Komplexität der einzelnen Teilsysteme ist ebenfalls beschränkt.*“ (Zitat DIN EN 61508-7:2011 [3])

Diese Prinzipien sind bei der Anwendung einer SPS-Sprache auf einer Sicherheits-SPS und passender Programmierrichtlinie grundsätzlich gegeben. Die Matrixmethode des IFA mit ihrer dreistufigen Struktur und der Verwendung von kleinen Softwaremodulen (Funktionsbausteinen) fördert die Modularisierung und strukturierte Programmierung. Die Verarbeitungsfunktionen (oder Ansteuerlogik, siehe Abschnitt 6.1) sollten dann auch in einzelne Module, jeweils auf ein Antriebselement bezogen, zerlegt werden (wie in Abbildung 9).

Abbildung 9:
Entwurf eines Softwarebeispiels (nach DIN EN ISO 13849-1, Anhang J)




## 5.9 Trennung von sicherheitsbezogener und nicht sicherheitsbezogener Software

Eng verknüpft mit der Struktur der sicherheitsbezogenen Anwendungssoftware ist auch deren Trennung von der nicht sicherheitsbezogenen, funktionalen Prozesssoftware. Letztere wird eventuell öfter modifiziert und dies soll ohne gefahrbringende Rückwirkungen auf die Sicherheitsfunktionen (und ggf. erneuter Validierung oder Zertifizierung der SRASW) möglich sein. Beim Einsatz zertifizierter Sicherheitssteuerungen ist diese Trennung typischerweise gegeben. Die folgenden Ausführungen beziehen sich daher eher auf die Anwendung von Standardkomponenten.

DIN EN ISO 13849-1 [1] verweist in Abschnitt 4.6.3 d) auf folgende zusätzliche fehlervermeidende Maßnahme:

„*d) Wo SRASW und nicht-SRASW in einer Komponente kombiniert werden:*

*1) SRASW und nicht-SRASW müssen in unterschiedlichen Funktionsblöcken codiert werden, mit sorgfältig definierten Datenschnittstellen.*

*2) Es darf keine logische Verknüpfung von nicht sicherheitsbezogenen und sicherheitsbezogenen Daten geben, die zur Herabstufung der Integrität der sicherheitsbezogenen Signale führen könnten, z. B. Verknüpfen eines sicherheitsbezogenen und eines nicht sicherheitsbezogenen Signals durch ein logisches „ODER“, dessen Ausgang sicherheitsbezogene Signale steuert.*“

Diese Anforderungen zielen darauf ab, dass Softwaremodule unabhängig (bzw. getrennt) voneinander ausgeführt werden müssen, sowohl im räumlichen als auch im zeitlichen Bereich.

Zeitlicher Bereich bedeutet: Ein Modul darf die Funktion eines anderen Moduls nicht durch zu hohe Beanspruchung der verfügbaren Rechenzeit beeinträchtigen. Es darf auch nicht die Ausführung des anderen Moduls durch Blockieren gemeinsam genutzter Ressourcen (Speicher, Semaphoren, Kommunikation usw.) verhindern. Dies muss durch angemessene Priorisierung der sicherheitsbezogenen Module, ausreichende Ressourcen, statische Speicherzuweisung usw. sichergestellt und gegebenenfalls durch fehlerbeherrschende Maßnahmen (z. B. Zykluszeitüberwachung) überwacht werden.

Räumlicher Bereich bedeutet: Die verwendeten Daten eines sicherheitsbezogenen Moduls dürfen nicht durch ein anderes Modul unzulässig verändert werden, hier insbesondere nicht durch ein nicht sicherheitsbezogenes Modul.

Wann liegt nun Trennung im räumlichen Bereich vor? Was sind typische Techniken? Nicht immer ist eine komplette Trennung möglich, z. B. wenn aus der Prozesssoftware Einschaltsignale für Aktoren in der SRASW freigegeben werden müssen. Für eine Kopplung von Softwaremodulen, die eigentlich unabhängig sein sollen, sind nur wenige Kopplungsarten geeignet[2]. Das sicherheitsbezogene Modul muss dabei immer die volle Kontrolle über gefahrbringende Aktoren behalten:

- Schnittstellenkopplung: Ein Zugriff auf das sicherheitsbezogene Modul oder seine Daten erfolgt nur über dafür implementierte Unterprogramme dieses Moduls (z. B. das Anfordern zum Einschalten eines Antriebs). Allein das sicherheitsbezogene Modul entscheidet, wie darauf zu reagieren ist.

[2] Weiterführende Hinweise siehe DIN EN 61508-3 (VDE 0803-3):2011 [18] im informativen Anhang F „Verfahren zum Erreichen der Nicht-Beeinflussung zwischen Softwareelementen auf einem einzelnen Rechner“

- Datenkopplung über eine Parameterliste: Der Zugriff auf das sicherheitsbezogene Modul oder seine Daten erfolgt nur über Variablen als Parameter eines Unterprogramms. Jedes Schreiben und Lesen einer Variablen ist für das sicherheitsbezogene Modul erkennbar und kann von ihm weiterverarbeitet werden, ohne zu gefahrbringenden Situationen zu führen.

Nicht empfohlen, nur in Ausnahmefällen und in sehr beschränkter Weise nach Programmierrichtlinien sind folgende Kopplungen zu verwenden:

- Kopplung über globale Daten: Sicherheitsbezogene Module verwenden globale Daten, auf die andere Module auch direkt zugreifen können. Es kann schwierig sein, das Zusammenwirken der Module zu verstehen und die Auswirkungen von Änderungen am Code einzuschätzen.

- Kontrollkopplung: Kopplung, die dem nicht sicherheitsbezogenen Modul eine unmittelbare Kontrolle über das sicherheitsbezogene Modul gibt, z. B. durch Übertragung eines Bits, das einen gefahrbringenden Antrieb einschalten kann. Dazu die konkrete Anforderung der DIN EN ISO 13849-1: *„kein Verknüpfen eines sicherheitsbezogenen und eines nicht sicherheitsbezogenen Signals durch ein logisches „ODER“, dessen Ausgang sicherheitsbezogene Signale steuert.“*

Mit einer inhaltlichen Kopplung zweier Module sind diese nicht mehr unabhängig. Beide Module müssen verstanden werden, um Codeänderungen bewerten zu können. Damit ist z. B. gemeint, dass

- von einem Modul direkt in ein anderes Modul gesprungen wird,

- ein Modul die Verzweigungsziele eines anderen Moduls beeinflusst,

- ein Modul direkt auf die Daten eines anderen Moduls zugreifen kann.

Eine richtige Kopplung und die Modultrennung kann nur durch das Lesen und Verstehen des Codes beurteilt werden und ist im Rahmen des Codereviews (Abschnitt 6.8) zu dokumentieren. Lässt sich nicht begründen, dass eine ausreichende Trennung oder Unabhängigkeit vom nicht sicherheitsbezogenen Modul vorliegt, dann ist dieses Modul ebenfalls als SRASW zu betrachten und in den Entwicklungsprozess einzubeziehen.

## 5.10 Funktionaler Test und erweiterter Test

Die Anforderungen der DIN EN ISO 13849 Teil 1 und 2 zu SRASW unterscheiden beim Aspekt der Validierung zwischen Funktionalen Tests (als Basismaßnahme) und erweitertem Funktionstest (als zusätzliche Maßnahme). Was ist der Unterschied?

Bei einem funktionalen Test werden die Sicherheitsfunktionen, wie spezifiziert, überprüft. Es wird z. B. eine Schutzeinrichtung ausgelöst und überprüft, ob der richtige Antrieb stoppt. Fehler in der Peripherie oder in der Software werden dabei nicht unterstellt oder eingebaut. Dies ist sinnvoll, weil diese Basismaßnahme für $PL_r$ a und b gefordert wird und diese $PL_r$ meist mit ungetesteten Architekturen der Kategorie B realisiert werden. Wo kein Test programmiert wurde, kann er auch nicht getestet werden.

Als zusätzliche Maßnahme zu den funktionalen Tests werden für höhere $PL_r$ c bis e die erweiterten Funktionstests gefordert. Diese $PL_r$ sind mittels programmierbarer Systeme nur durch Kategorie 2, 3 und 4 mit Diagnosefunktionen zu erreichen. Dabei sind Diagnosefunktionen meist auch in der SRASW realisiert (direkte/indirekte Überwachung, Kreuzvergleich, Plausibilisierung, usw.). Zur Definition aus der Grund-Sicherheitsnorm DIN EN 61508-7:2011 [3]:

*„Ziel [der Erweiterten Funktionstests]: Aufdeckung von Ausfällen während der Spezifikations-, Entwurfs- und Entwicklungsphase. Test des Verhaltens des sicherheitsbezogenen Systems im Falle seltener oder nicht festgelegter Eingaben.*

*Beschreibung: Die erweiterte Funktionsprüfung überprüft das funktionale Verhalten des sicherheitsbezogenen Systems als Reaktion auf Eingabebedingungen, die nur selten erwartet werden (zum Beispiel mehrfacher Ausfall) oder die außerhalb der Spezifikation des sicherheitsbezogenen Systems liegen (zum Beispiel nicht ordnungsgemäßer Betrieb). Für seltene Bedingungen wird das beobachtete Verhalten des sicherheitsbezogenen Systems mit der Spezifikation verglichen. Wenn die Reaktion des sicherheitsbezogenen Systems nicht festgelegt ist, sollte getestet werden, ob die Sicherheit der Anlage bei der beobachteten Reaktion erhalten bleibt.“*

Beim erweiterten Funktionstest werden typischerweise Fehler in den Peripheriegeräten der Steuerung oder der Steuerung selbst eingebaut bzw. simuliert. Damit können die Diagnosefunktionen in der SRASW, die in den Eingangs- und Ausgangsblöcken (Abbildung 8) realisiert wurden, sowie deren Fehlerreaktion getestet werden. Problematisch kann es dabei sein zu erkennen, ob wirklich die zu testende Diagnosefunktion der SRASW – oder doch die eingebettete Software – den sicheren Zustand der Steuerung herbeigeführt hat.

Für $PL_r$ d und e wird empfohlen, zusätzlich erweiterte Testfälle, die auf Grenzwertanalysen beruhen, durchzuführen. Dies ist meist nur sinnvoll, wenn die Eingangswerte analog sind und unterschiedliche Bereiche dieser Eingangswerte gebildet werden. Ein Beispiel: Ein Temperatursensor liefert Werte zwischen -20 und +100 °C. Je nach Temperaturbereich werden entsprechend Ausgänge gesteuert. Es werden drei Bereiche spezifiziert:

- „zu kalt“: -20 bis +9,99 °C

- „Temperatur OK“: +10 bis +29,99 °C

- „zu warm“: +30 bis +100 °C

Die Grenzwertanalyse dieser Spezifikation liefert die Extremwerte -20 und 100 °C sowie die Bereichsgrenzen 10 und 30 °C. Dann soll mit Eingangswerten an bzw. um diese Grenzen und

Extremwerte herum getestet werden, da dort Fehler in der Verarbeitung vermutet werden können.

Dabei sollten ggf. auch zeitlich grenzwertige Situationen überprüft werden, wenn z. B. eine Zeitüberwachung mit grenzwertigen Verzögerungen, nahe bei der nominellen Überwachungszeit, getestet wird.

Die Matrixmethode erlaubt, neben den funktionalen Tests der Sicherheitsfunktionen auch weitere Testfälle entsprechend der oben genannten Methoden zu definieren und zu validieren.

## 5.11 Testabdeckung

Wie viel und wie lange muss getestet werden? Es geht um das Kriterium der Testabdeckung, manchmal auch Testüberdeckung genannt. Sie ist ein wichtiges Maß zur Softwarequalität. Die Testabdeckung gibt an, welcher Anteil der Software durch die Gesamtheit der Testfälle ausgeführt wird. Mit einer höheren Anzahl von sinnvoll ausgewählten Testfällen kann die Testabdeckung und damit die Softwarequalität verbessert werden. Allerdings erhöht sich dadurch auch der Testaufwand. Ideal wäre eine 100%ige Testabdeckung aller Softwareelemente. In der Praxis orientiert sich der erforderliche Testaufwand an der erforderlichen Sicherheitsintegrität, dem $PL_r$.

Die Grund-Sicherheitsnorm DIN EN 61508-3:2011 [3] gibt zu diesem Aspekt in der Tabelle B.2 sehr konkret Hilfestellung und soll hier für eine Interpretation in Bezug auf die Matrixmethode des IFA genutzt werden (Tabelle 3). Es handelt sich hier zwar um Testmethoden, die Kenntnisse über den Programmcode erfordern und damit nicht dem in DIN EN ISO 13849 geforderten Black-Box-Testen entsprechen. Dennoch lassen sich Rückschlüsse auf eine erreichbare Testabdeckung für die Verarbeitung in der SRASW (Abbildung 8) ziehen, je nach Umfang der Testfälle.

Tabelle 3:
Empfehlungen für Testabdeckung

| Testabdeckung in Anlehnung an IEC 61508-3 | $PL_r$ SIL | Umfang der Testfälle im Black-Box-Testen der SRASW |
|---|---|---|
| 100 % der Eingänge | a, b und c SIL1 | Mindestens alle sicherheitsrelevanten Eingänge, d. h. alle Schutzeinrichtungen und damit die Funktionsbausteine der Vorverarbeitungsebene einmal anfordern und den Wiederanlauf testen. Ziel: Stellt sicher, dass jedes Unterprogramm der SRASW, auch die Funktionsbausteine der Ansteuerlogik und Ansteuerungsebene, mindestens einmal aufgerufen worden ist. Empfohlen wird aber auch für diese $PL_r$ der Testumfang wie in der nächsten Zeile für $PL_r$ d. |
| 100 % der Programmanweisungen | d SIL2 | Alle Sicherheitsfunktionen in allen Betriebsarten anfordern und den Wiederanlauf testen. Ziel: Stellt sicher, dass alle Anweisungen in der Ansteuerlogik der SRASW (Abbildung 7) mindestens einmal ausgeführt worden sind. Enthält $PL_r$ a, b, c. |
| 100 % der Programmverzweigungen | e SIL3 | Alle Sicherheitsfunktionen in allen Betriebsarten anfordern, den Wiederanlauf testen und alle Diagnosefunktionen durch Fehlersimulation testen. Ziel: Beide Möglichkeiten jeder Verzweigung in der Ansteuerlogik der SRASW sollten getestet werden. Enthält $PL_r$ a, b, c, d. |

Zertifizierte Herstellerfunktionsbausteine und bereits validierte Funktionsbausteine müssen nicht mehr getestet werden, wohl aber deren Verschaltung und Parametrierung. Noch nicht validierte Funktionsbausteine müssen separat im Rahmen der Modulentwicklung getestet und validiert werden (siehe Abschnitt 5.10).

Die erreichte Testabdeckung kann bei der Matrixmethode des IFA und in SOFTEMA in den Prüfspalten der Matrixdarstellungen (B4, B5) dokumentiert werden.

## 5.12 Dokumentation

Bevor der Hersteller die EG-Konformitätserklärung für eine Maschine ausstellt, muss er eine technische Dokumentation ausarbeiten. Dies darf aber nicht nur als ungeliebte Pflicht verstanden werden. Eine maßvolle interne Dokumentation bietet auch Vorteile für die eigene nachhaltige Entwicklungsarbeit und dient der Absicherung bei eventuellen rechtlichen Konflikten. Was sagt DIN EN ISO 13849-1 in Bezug auf SRASW dazu? Knapp, aber eindeutig:

*„Alle Lebenszyklus- und Änderungsaktivitäten müssen dokumentiert werden. Die Dokumentation muss vollständig, verfügbar, lesbar und verständlich sein.“*

Die Matrixmethode und das Tool SOFTEMA des IFA unterstützen eine ausreichende Entwicklungsdokumentation, wie in Kapitel 6 und mit den vielen Beispielen in Kapitel 7 dargestellt. Im Kapitel 13 werden die Themen „Technische Dokumentation“ und „Benutzerinformation“ gesondert betrachtet.

## 5.13 Konfigurationsmanagement

Was ist mit Konfigurationsmanagement gemeint? Besonders bei sicherheitsbezogener Software ist selbstverständlich und daher zu fordern, dass deren Entwicklung für alle Beteiligten und spätere Prüfungen nachvollzogen werden kann:

- Wer hat wann spezifiziert, programmiert, in Betrieb genommen, verifiziert, validiert?
- Womit wurde entwickelt, z. B. Werkzeuge und ihre Einstellungen, wieder verwendete Softwaremodule und ihre Identifikation, Programmierrichtlinie?
- Welches sind die passenden Handbücher für die Entwicklungswerkzeuge?
- Welche Programmversionen sind in welchen SRP/CS geladen?

Diese und weitere notwendige Informationen sowie alle relevanten Entwicklungsdokumente sind für eine spätere Nutzung – z. B. bei einer Modifikation nach fünf Jahren Betrieb – zu dokumentieren. Mit der Matrixmethode und dem Tool SOFTEMA werden alle diese Informationen in einem Standardformat (Microsoft Excel) erfasst und archiviert. Somit kann bei entsprechend dauerhafter Archivierung noch Jahre später auf die Daten zugegriffen werden.

## 5.14 Modifikationen

Erfahrungsgemäß wird auch eine zunächst getestete SRASW noch während der Inbetriebnahme einer Anlage/Maschine eifrig erweitert und angepasst. Diesen Vorgang nennt man „Modifikation“. Oft gehen diese Änderungen so weit, dass nicht nur die Codierung, sondern auch die ursprüngliche Spezifikation nicht mehr passt: Sie müsste eigentlich überarbeitet werden. Durch geänderte Sicherheitsfunktionen an einer Stelle der Anlage/Maschine können auch die anderen, zunächst nicht modifizierten Sicherheitsfunktionen betroffen sein. Oder es ergeben sich durch die Modifikationen Lücken im Sicherheitskonzept. Dies gilt es zu überprüfen und gegebenenfalls die notwendigen Phasen des V-Modells zu wiederholen.

Die Praxis zeigt aber, dass auch an einer installierten Anlage/Maschine gelegentlich ein Not-Halt oder eine Schutztür ergänzt werden muss. Oft wird auch der Bearbeitungsprozess optimiert: Das Sicherheitskonzept ist ebenfalls anzupassen. Die existierende Software muss „modifiziert“ werden. Wohl gemerkt: bei SRP/CS, die schon länger und meist ohne durch Softwarefehler bedingte Ausfälle betrieben wurden – was auch bedeuten könnte, dass ein vorhandener „versteckter“ Fehler nur noch nicht wirksam wurde. Dies kann sich aber nach einer Modifikation ändern, wenn die Software z. B. nicht ausreichend strukturiert wurde und einzelne Module/Funktionen somit untereinander nicht vollständig rückwirkungsfrei sind.

In den beschriebenen Situationen zeigt sich oft Murphys Gesetz: Das Programm wurde schon vor etlichen Jahren geschrieben, die ursprünglich Programmierenden haben dringendere Aufgaben oder sind bereits in einem anderen Unternehmen tätig. Hier zahlt es sich für die Sicherheit, aber auch Verfügbarkeit der Maschine aus, wenn die Software die oben genannten Merkmale aufweist: Lesbarkeit, Struktur, Verständlichkeit und auch das Merkmal, einfach und fehlervermeidend modifiziert werden zu können – unabhängig von den jeweils verfügbaren Programmierenden.

Im Prinzip muss man nach einer Modifikation wieder dort im Entwicklungsprozess, also im V-Modell (Abbildung 5), einsteigen, wo etwas geändert wurde, z. B.:

- Bei geänderter Codierung sind evtl. das Codereview, der Integrationstest sowie die Validierung erneut durchzuführen.
- Musste gar die Spezifikation geändert werden, ist diese ebenfalls erneut zu verifizieren, z. B. durch Review (Gegenlesen) einer anderen Person, damit sich keine Fehler an anderer Stelle der Spezifikation einschleichen. Dementsprechend müssen alle Entwicklungs- und Verifikationsmaßnahmen sowie die Validierung der betroffenen Sicherheitsfunktionen wiederholt werden.

Die Matrixmethode des IFA unterstützt diese Modifikationen (Abschnitt 6.15) und SOFTEMA automatisiert die entsprechenden Änderungen in den Entwicklungsdokumenten.

## 5.15 Vier-Augen-Prinzip und Unabhängigkeitsgrade

Überprüfende Aktivitäten wie Verifikation und Validierung sollen die Konformität der Gestaltung der SRP/CS mit der Maschinenrichtlinie sicherstellen. Diese Aktivitäten sollten so früh wie möglich während der Entwicklung begonnen werden, sodass Fehler rechtzeitig erkannt und behoben werden können. Wer Programmcode schreibt, sollte aber nicht selbst seine Ergebnisse überprüfen. Hier kommt das Vier-Augen-Prinzip (englisch: Two-man rule) ins Spiel. In der Praxis wird oft gefragt: Wann ist das Vier-Augen-Prinzip überhaupt notwendig? Wie unabhängig muss die zweite Person sein? Kann das jemand aus meiner Arbeitsgruppe machen?

DIN EN ISO 13849-2:2013 gibt in Abschnitt 4.1 die Empfehlung: „*Die Validierung sollte von Personen durchgeführt werden, die unabhängig von der Gestaltung der SRP/CS sind.*“ Dies können unabhängige Personen, Personen aus unabhängigen Abteilungen oder unabhängigen Organisationen sein (Definitionen in Tabelle 4). Zusätzlich wird in diesem Report der Begriff „andere Person“ eingeführt. Das meint einfach eine andere Person als diejenige, deren Ergebnis (Spezifikation, Code etc.) überprüft werden soll.

Der Grad der Unabhängigkeit sollte dabei dem Risiko, also dem erforderlichen Performance Level $PL_r$, angemessen sein. Teil 2 der Norm ergänzt in einer Anmerkung: „*Unabhängige Person bedeutet nicht unbedingt, dass eine Prüfung durch Dritte erforderlich ist.*“ Sofern nicht durch gesetzliche Regelungen vorgeschrieben, bleibt die Prüfung also typischerweise „im Haus“. Externe Prüfstellen werden dennoch gerne für Beratung und Bewertung von technischen und organisationalen Maßnahmen hinzugezogen. Tabelle 4 zeigt die Empfehlung des IFA zu den Unabhängigkeitsgraden.

Dazu ein Zitat aus DIN EN 61508-1 (VDE 0803-1) [3], Abschnitt 8.2.16, Anmerkung 1:

„*Abhängig von der Organisation und der Erfahrung innerhalb des Unternehmens kann es notwendig sein, die Anforderung in Bezug auf unabhängige Personen und Abteilungen durch eine externe Organisation zu erfüllen. Andererseits können Unternehmen, die interne, in der Risikobeurteilung und Anwendung von sicherheitsbezogenen Systemen erfahrene Institutionen besitzen, die unabhängig und getrennt (in Bezug auf Management und andere Ressourcen) von den für die Hauptentwicklung Verantwortlichen sind, ihre eigenen Ressourcen verwenden, um die Anforderungen für eine unabhängige Organisation zu erfüllen.*“

Tabelle 4:
Definition der Unabhängigkeitsgrade für SRASW (angelehnt an DIN EN 61508-4:2011)

| | Definition (Beispiel) als Empfehlung des IFA |
|---|---|
| Andere Person | Person, die nicht die zu überprüfende Tätigkeiten selbst durchgeführt hat, aber in das Projekt eingebunden sein kann oder Verantwortung trägt (ja: Projektleitung, Vorgesetzte, Beteiligte an der betrachteten Tätigkeit) |
| Unabhängige Person | Person, die nicht eingebunden ist in die zu überprüfende Tätigkeiten und die keine direkte Verantwortung für diese Tätigkeiten trägt (nein: Projektleitung, Vorgesetzte, Beteiligte an der betrachteten Tätigkeit; ja: Hardwareprojektierende für SRASW-Entwicklung, Inbetriebnehmende für Projektierungstätigkeit) |
| Unabhängige Abteilung | Abteilung, die nicht in Verbindung mit den Projekt-/Entwicklungsabteilungen steht, die verantwortlich für die SRASW-Tätigkeiten sind (Person aus der Qualitätsabteilung/der CE-Abteilung/dem Schaltanlagenbau usw.) |
| Unabhängige Organisation | Organisation, die aufgrund ihres Managements und ihrer anderen Mittel nicht in Verbindung mit den Entwicklungsorganisationen steht (anderer Geschäftsbereich, anderes Unternehmen, Prüfstelle) |

Mit diesen Definitionen wird in Tabelle 5 vom IFA eine detaillierte Empfehlung für den Grad der minimalen Unabhängigkeit gegeben, der vom maßgeblichen $PL_r$ der SRASW-Entwicklung abhängt. Dies geschieht in Anlehnung an die Grund-Sicherheitsnorm DIN EN 61508-1:2011, Tabelle 5. Dabei wird aber berücksichtigt, dass die Sicherheitsfunktionen im Anwendungsbereich der DIN EN ISO 13849-1 typischerweise eine geringere Komplexität und damit eine geringere Fehlerwahrscheinlichkeit aufweisen als die Sicherheitsfunktionen im typischen Anwendungsbereich der DIN EN 61508-1. Daher wird der Grad der notwendigen Unabhängigkeit für SRASW in Tabelle 5 gegenüber der DIN EN 61508-1 um eine Stufe herabgesetzt.

Tabelle 5:
Grad der Unabhängigkeit für Verifikation und Validierung (SRASW)

| Minimaler Unabhängigkeitsgrad für Verifikation und Validierung bei SRASW * als Empfehlung des IFA | Maßgeblicher $PL_r$ für SRASW-Entwicklung * | | | |
|---|---|---|---|---|
| | a und b | c | d | e |
| Andere Person** | Für alle SRASW empfohlen | Für alle SRASW | Standard-SRASW | Nicht ausreichend |
| Unabhängige Person | Möglich, aber nicht notwendig | Möglich, aber nicht notwendig | Komplex oder neue SRASW | Standard-SRASW |
| Unabhängige Abteilung | Möglich, aber nicht notwendig | Möglich, aber nicht notwendig | Möglich, aber nicht notwendig | Komplex oder neue SRASW |
| Unabhängige Organisation*** | Möglich, aber nicht notwendig | Möglich, aber nicht notwendig | Möglich, aber nicht notwendig | Möglich, aber nicht notwendig |

* In Anlehnung an die Grund-Sicherheitsnorm DIN EN 61508-1 [3], Tabelle 5

** Niedrigster Unabhängigkeitsgrad

*** Höchster Unabhängigkeitsgrad

Die grünen Tabellenfelder in Tabelle 5 beziehen sich auf die Entwicklung von SRASW mit SPS-Sprachen auf einer Sicherheits-SPS, wenn Erfahrung mit der Art des Entwurfs sowie der eingesetzten Technologie vorhanden ist und die Programme keinen höheren Grad der Komplexität aufweisen – also für die typischen Anwender der Matrixmethode. Das bedeutet für die in diesem IFA Report betrachtete „Standard-SRASW“: Für $PL_r$ c und d genügt eine andere Person für das Vier-Augen-Prinzip, bei $PL_r$ e ist eine unabhängige Person empfohlen. Man sollte bedenken, dass schon bei $PL_r$ c im Fehlerfalle Menschenleben gefährdet sein können. Auch schon bei $PL_r$ a und b wird das Vier-Augen-Prinzip empfohlen.

Die gelben Tabellenfelder beziehen sich auf alle anderen fehlerträchtigen Situationen, in denen z. B. sehr große bzw. sehr komplexe Programme entwickelt oder eine neuartige Softwarearchitektur oder eine neuartige Technologie (FVL-Sprachen, bisher unbekannte Steuerungen, selbst entwickelte Softwarearchitektur usw.) eingesetzt werden. In diesen Situationen wird meist auch die Matrixmethode des IFA nicht eingesetzt werden können.

Die grauen Tabellenfelder stehen für nicht notwendige, aber mögliche Unabhängigkeitsgrade. Wird ein niedrigerer Unabhängigkeitsgrad als in Tabelle 5 gewählt, sollte dies begründet werden.

## 5.16 Projektmanagement

Bei der Entwicklung von SRESW (Embedded-Software) ab $PL_r$ c fordert DIN EN ISO 13849-1 ein „Projektmanagement- und Qualitätsmanagementsystem vergleichbar mit z. B. der Reihe IEC 61508 oder ISO 9001“. Ein Projektmanagement im Unternehmen wird im Zusammenhang der Anforderungen an SRASW dagegen nicht explizit gefordert. Immerhin wird das V-Modell als Ablauforganisation vorgegeben. An anderer relevanter Stelle wird das Projektmanagement – nicht näher spezifiziert – empfohlen: im Anhang G.4 „Maßnahmen zur Vermeidung systematischer Ausfälle während der Integration des SRP/CS“. Aber auch hier steht die SRASW nicht direkt im Fokus.

Daraus lässt sich ableiten, dass in Bezug auf SRASW die Umsetzung des V-Modells als Entwicklungslebenszyklus mit Verifikation und Validierung (Abschnitt 5.2) unter Beachtung des Vier-Augen-Prinzips (Abschnitt 5.15) zur Qualitätssicherung eine angemessene fehlervermeidende Maßnahme darstellt. Die Anwendung der Matrixmethode unterstützt diese Qualitätssicherung. Darüber hinaus wird ein „*Projektmanagement- und Qualitätsmanagementsystem vergleichbar mit z. B. der Reihe IEC 61508 oder ISO 9001*“ für SRASW nicht gefordert.

Die DIN EN ISO 13849-1 bietet auch keine eigene Definition von „Projektmanagement“ an, daher sei hier der Vollständigkeit wegen auf eine Beschreibung aus der Grund-Sicherheitsnorm DIN EN 61508-7 [3], Abschnitt B.1.1 verwiesen:

„*Ziel: Vermeidung von Ausfällen durch Annahme eines organisatorischen Modells sowie von Regeln und Maßnahmen für die Entwicklung und den Test sicherheitsbezogener Systeme.*

*Beschreibung: Die wichtigsten und besten Maßnahmen sind*

- *die Erstellung eines Organisationsmodells speziell für die Qualitätssicherung, welches in einem Qualitätssicherungshandbuch niedergelegt ist; und*
- *die Festlegung von Regeln und Maßnahmen für die Erstellung und Validierung von sicherheitsbezogenen Systemen in Projektplanung und projektspezifischen Richtlinien.*

*Mehrere wichtige Basisprinzipien werden im Folgenden genannt:*

- *Definition einer Entwurfsorganisation:*
  - *Aufgaben und Verantwortlichkeiten der organisatorischen Einheiten;*
  - *Befugnisse der Abteilung für die Qualitätssicherung;*
  - *Unabhängigkeit der Qualitätssicherung (interne Inspektion) von der Entwicklung;*
- *Definition des Ablaufplans (Phasenmodelle):*
  - *Bestimmung aller Tätigkeiten, die während der Durchführung des Projekts sachdienlich sind, einschließlich der internen Inspektionen und deren Zeitplanung;*
  - *Projektaktualisierung;*
- *Definition eines feststehenden Ablaufs einer internen Inspektion:*
  - *Planung, Ausführung und Überprüfung der Inspektion (Inspektionstheorie);*
  - *Freigabemechanismus für Teilprodukte;*
  - *Sicherung von Wiederholungsinspektionen;*
- *Konfigurationsmanagement:*
  - *Verwaltung und Kontrolle der Versionen;*
  - *Erkennung der Auswirkungen von Modifikationen;*
  - *Inspektion der Beibehaltung der Eigenschaften nach Modifikationen;*
- *Einführung einer quantitativen Beurteilung von Qualitätssicherungsmaßnahmen:*
  - *Festlegung der Anforderungen;*
  - *Ausfallstatistiken;*
- *Einführung von rechnerunterstützten universellen Methoden, Werkzeugen und Personalschulung.*“

## 5.17 Externe Prüfung von SRASW

Die Maschinenrichtlinie und das entsprechende nationale Produktsicherheitsgesetz verpflichten Hersteller, Importeur oder Händler von Maschinen, umfangreiche Anforderungen hinsichtlich der Arbeitssicherheit und des Gesundheitsschutzes einzuhalten. Werden die Anforderungen nicht eingehalten, kann dies zu weitgehenden Konsequenzen führen, z. B. zu Produkthaftungsfällen oder zu nachträglichen Forderungen von Aufsichtsbehörden.

Die Maschinenrichtlinie fordert, dass für die Konformitätsbewertung von besonders gefährlichen Maschinen bzw. sicherheitsrelevanten Bauteilen, die im Anhang IV gelistet sind, unter anderem eine europäisch notifizierte Prüf- und Zertifizierungsstelle beauftragt wird. Für die allermeisten Maschinen finden aber die externen Prüfungen bei einer Prüfstelle freiwillig statt und geben dadurch Gewissheit, dass die Produkte und die technischen Unterlagen den nationalen und europäischen Sicherheits- und Gesundheitsschutzanforderungen entsprechen.

Diese externen Prüfungen haben auch für Betreiber Vorteile. Wer kann beim Einkauf von Produkten schon genau beurteilen, wie es um die Sicherheit bestellt ist? Häufig muss beim Einkauf dem Hersteller vertraut werden. Das kann teuer werden, wenn sich später im betrieblichen Einsatz Sicherheitsmängel herausstellen. Geprüfte Produkte bieten eine Gewähr, ein sicherheitstechnisch einwandfreies Produkt zu kaufen – zum Vorteil der Beschäftigten und des Betriebs.

Das Prüf- und Zertifizierungssystem der Deutschen Gesetzlichen Unfallversicherung – DGUV Test – führt 16 Prüf- und Zertifizierungsstellen zusammen (www.dguv.de/dguv-test). Die meisten Stellen prüfen und zertifizieren Maschinen und damit auch sicherheitsbezogene Anwendungsprogramme. Die meisten Prüfungen müssen aufgrund der Komplexität der Programme schon während der Konzept- und Entwicklungsphasen mit den Prüfstellen abgestimmt werden. Das trifft auch auf die Art der Spezifikation, Dokumentation und Validierung zu, z. B. nach der Matrixmethode des IFA, die damit die Prüfung und Zertifizierung erleichtern kann.

# 6 Entwicklung von sicherheitsgerichteter Anwendungssoftware

Dieses Kapitel ist der Kern des vorliegenden Reports und im Wesentlichen dem Forschungsbericht des DGUV Projektes FF-FP0319 (siehe Abschnitt 2.2 dieses Reports) entnommen. Im Folgenden wird die Matrixmethode des IFA eingeführt und anhand von Beispielen dargestellt.

## 6.1 Matrixbasierte Spezifikation und Dokumentation

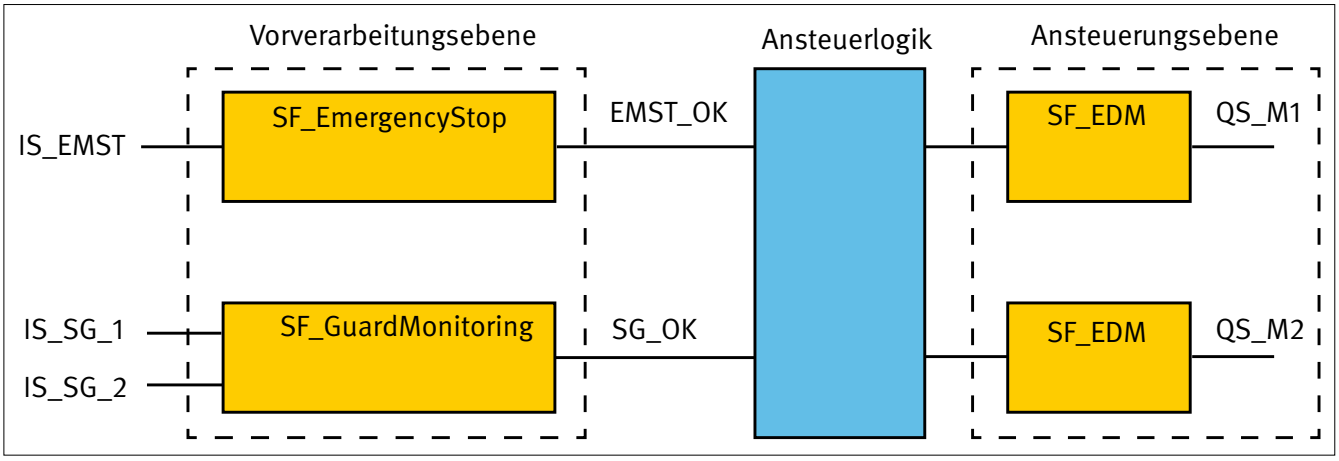
Die Matrixmethode des IFA setzt auf eine kompakte Spezifikation und Dokumentation von SRASW, die zusätzlich zu den üblichen Unterlagen, wie den Konstruktionsunterlagen einer Maschine oder dem Programmausdruck der SRASW, erstellt wird.

Bei dieser Methode wird eine bestimmte Struktur der Software vorausgesetzt. Diese Struktur ist auch wesentliche Voraussetzung für die Vereinfachung des V-Modells (siehe Abschnitt 5.4). Abbildung 10 zeigt beispielhaft die Softwarestruktur.

Man kann der Abbildung eine klare Dreiteilung entnehmen. Auf der linken Seite befindet sich die Vorverarbeitungsebene, in der die Eingänge der SPS vorwiegend mit Bibliotheksbausteinen, die von den Herstellern mitgeliefert werden, verarbeitet werden. Auf der rechten Seite ist die Ansteuerungsebene dargestellt, in der Ausgänge der SPS mit Bibliotheksbausteinen oder direkt angesteuert werden.

In der Mitte ist – blau dargestellt – die noch zu spezifizierende Ansteuerlogik der Aktoren. In diesem Beispiel werden in der Ansteuerlogik die Ausgänge EMST_OK und SG_OK der verwendeten Bibliotheksbausteine der Vorverarbeitungsebene logisch verschaltet. Diese Verschaltung besteht nur aus den Verknüpfungen UND, ODER und NICHT. Zeitliche Bedingungen müssen in der Vorverarbeitungs- und Ansteuerungsebene implementiert werden.

Abbildung 10:
Beispielhafte vorgegebene Softwarestruktur . Die Funktionsbausteine sind konform zu PLCopen [8].




Diese Gliederung entspricht dem Aufbauschema mit Sensor ⇢ Logik ⇢ Aktor für Sicherheitsfunktionen aus DIN EN ISO 13849-1 (siehe Abschnitt 5.7).

Ebenfalls wichtig für die Matrixmethode des IFA ist eine strukturierte Namensgebung für Variablen. Dies dient der besseren Übersicht im Programm und hilft, Fehler zu vermeiden bzw. schnell zu finden, da zusammengehörende Module auch immer gleiche Namensteile haben.

Die Matrixmethode besteht im Wesentlichen aus den folgenden Schritten:

1. Definition der Sicherheitsfunktionen (Dokument A1),
2. Auflistung der Variablennamen und Adressen von sicherheitsrelevanten Ein- und Ausgängen in der I/O-Liste (Dokument A2.4),
3. Auswahl der fehlervermeidenden Maßnahmen aus dem Katalog (Dokument A3),
4. Festlegung der normativen Anforderungen (Dokument A4),
5. Festlegung der Architektur des Sicherheitsprogramms (Dokument B1),
6. Festlegung der Modularchitektur mit Bausteinen der Vorverarbeitungs- und Ansteuerungsebene (Dokument B3),

7. Aufstellung der Matrix mit den Eintragungen für die Softwarespezifikation (Dokument B4),

8. Verifikation der Dokumente B1, B3 und B4 gegen die Spezifikation der Sicherheitsfunktionen (Dokument V1),

9. Programmierung der Software,

10. Codereview (Dokument C1), möglichst durch eine zweite Person,

11. Validierung der Software, d. h. Analyse und Funktionstest (Dokument D1), möglichst durch eine zweite Person,

12. Archivieren der Software und Dokumentation sowie sämtlicher benötigter Prüfunterlagen.

Anmerkung: Die hier dargestellte Matrixmethode kann auch zur kompakten Darstellung von nicht sicherheitsrelevanten Schaltvorgängen genutzt werden.

Die folgenden Kapitel beschreiben in mehreren aufeinander aufbauenden Beispielen die Anwendung der Matrixmethode des IFA. Die Beispiele sollen einen „roten Faden“ für die Anwendung darstellen. Bei der Anwendung der Methode kann sie nach den jeweiligen Erfordernissen angepasst werden. Die Inhalte einiger Dokumente (z. B. A3) sind stark projektabhängig. Die Anwendung der Methode wird durch das in Kapitel 14 beschriebene Tool SOFTEMA unterstützt.

Bei der Darstellung der Beispiele ist bewusst auf Neutralität bezüglich der Steuerungshersteller geachtet worden.

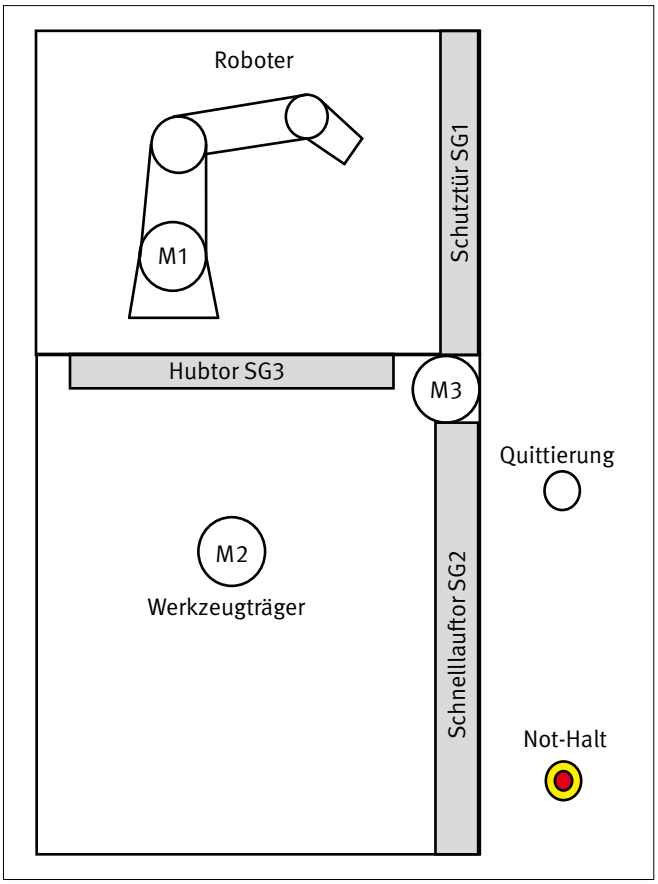
## 6.2 Beispiel zur matrixbasierten Spezifikation und Dokumentation

Zur kompakten und übersichtlichen Dokumentation der hier betrachteten Beispiele wurden für jedes Beispiel alle zugehörigen Dokumente in einer Excel-Mappe zusammengefasst. Der Leser kann dann zu jedem Dokument bequem durch Anklicken der beschrifteten „Reiter“ der Tabellenblätter navigieren. Diese Excel-Dokumente sind in einer einzigen Archivdatei im Downloadbereich dieses IFA Reports zu finden.

An einem überschaubaren Beispiel wird der komplette Aufbau der Excel-Mappe erläutert. Die Beispiele zum Download (Kapitel 7) sind in einem gegenüber dem Forschungsprojekt aktualisierten Format verfügbar. Dieses Format ist für die automatisierte Bearbeitung durch ein Softwaretool wie SOFTEMA besser geeignet.

Als Beispiel zur Erläuterung der Matrixmethode dient in diesem Abschnitt eine Roboterfertigungszelle (entspricht Beispiel in Abschnitt 7.1), wie sie Abbildung 11 zeigt. Die Abbildung stellt die Anlagenskizze (Dokument A2.1) dar. Dieses Dokument ist als optional in Tabelle 1 (Seite 23) gekennzeichnet. Mit dieser Anlagenskizze lässt sich das Beispiel allerdings einfacher erklären.

Abbildung 11:
Anlagenskizze (Dokument A2.1), Beispiel Roboterfertigungszelle




Die Funktion der Fertigungszelle lässt sich wie folgt beschreiben:

- Der Roboter (M1) bestückt ein Werkzeug auf dem Werkzeugträger mit Material. Dazu greift er durch das automatische Hubtor SG3.
- Nach dem Bestücken zieht sich der Roboter zurück und das Hubtor schließt sich automatisch wieder.
- Nach einer Aushärtezeit öffnet eine Bedienperson das Schnelllauftor SG2 (Motor M3) und entnimmt das fertige Teil aus dem Werkzeug, reinigt das Werkzeug und verschließt danach das Schnelllauftor wieder.
- Sobald das Schnelllauftor geschlossen ist, kann der Roboter das Werkzeug wieder bestücken.
- Das Schnelllauftor SG2 besitzt zur Vermeidung von Quetschungen an der Schließkante eine Sicherheitsleiste SL_SG2 (nicht dargestellt).
- Die Schutztür SG1 dient als Wartungszugang zum Roboter.
- Im Bild sind als Befehlsgeräte nur der Not-Halt EMST und die Quittierung dargestellt, da nur sie sicherheitsrelevant sind.
- Für diese Fertigungszelle gibt es nur die Betriebsart Automatikbetrieb.

## 6.3 Spezifikation der Sicherheitsfunktionen

Aus der Risikobeurteilung ergeben sich hier fünf Sicherheitsfunktionen. Die ausführliche Definition einer Sicherheitsfunktion in der Sicherheitsspezifikation zeigt Tabelle 6 am Beispiel des Not-Halts. Der Aufbau der Definition orientiert sich am SISTEMA-Kochbuch 6 [14] und an den Angaben im Kasten 6.1 des BGIA-Reports 2/2008 [2].

Für das Softwareengineering aller Sicherheitsfunktionen des Beispiels sind diese kompakt in Tabelle 7 (Dokument A1) abgebildet.

Tabelle 6:
Definition der Sicherheitsfunktion Not-Halt

| | SF1 | Not-Halt |
|---|---|---|
| 1 | Beschreibung | Wenn der Not-Halt-Taster EMST betätigt wird, werden die Antriebe M1, M2, M3 gestoppt. |
| 2 | Auslösendes Ereignis | Betätigen des Not-Halt-Tasters |
| 3 | Sicherheitsgerichtete Reaktion | Stillsetzen: Drehzahl = 0, Stoppkategorie 0 (STO) |
| 4 | Gefahrbringendes Maschinenteil | Achsen M1, M2 und M3 |
| 5 | Reaktion bei Fehler der SF | Stillsetzen: Drehzahl = 0, Stoppkategorie 0 (STO) |
| 6 | $PL_r$ | d |
| 7 | Betriebsart | Alle Betriebsarten |
| 8 | Parameter/Fehler | 8.1 Parameter: Diskrepanzzeit $T_{dis}$ = 50 ms für die Kontakte des Not-Halt-Tasters -› Fehler |
| | | 8.2 Parameter: Diskrepanzzeit $T_{dis\_}$ Q = 1 s für Schützüberwachung |
| 9 | Nachlauf | 100 ms |
| 10 | Verhalten bei Energieausfall | Stillsetzen: Drehzahl = 0, Stoppkategorie 0 |
| 11 | Wiederanlaufbedingungen | Keine Fehler, Not-Halt-Taster geschlossen, Quittiertaster betätigt |
| 12 | Priorität | 1 (höchste Priorität) |
| **Version** | | |
| **Datum** | | |
| **Name** | | |

Tabelle 7:
Spezifikation der Sicherheitsfunktionen (Dokument A1)

| Nr. | Beschreibung | $PL_r$ | Reaktionszeit in ms | Priorität |
|---|---|---|---|---|
| SF1 | Wenn der Not-Halt EMST betätigt wird, werden M1, M2 und M3 abgeschaltet. | d | 100 | 1 |
| SF2 | Wenn Schutztür SG1 auf, dann wird M1 abgeschaltet. | d | 100 | 2 |
| SF3 | Wenn Schnelllauftor SG2 auf, dann wird M2 abgeschaltet. | d | 100 | 2 |
| SF4 | Wenn Schnelllauftor SG2 auf und Hubtor SG3 auf, dann wird M1 abgeschaltet. | d | 100 | 2 |
| SF5 | Wenn die Sicherheitsleiste SL_SG2 des Schnelllauftors SG2 betätigt wird, dann wird M3 abgeschaltet. | d | 100 | 2 |

In dieser kompakten Fassung sind einige Angaben weggelassen worden. Das auslösende Ereignis sowie die Reaktion der Sicherheitsfunktionen sind in der Spalte Beschreibung dargestellt. Die übrigen Parameter aus Tabelle 6 sind übergeordnet in den Katalog der fehlervermeidenden Maßnahmen A3 aufgenommen worden. Damit wird die Übersichtlichkeit in diesem Dokument erhöht. Die Betriebsart ist in diesem Beispiel nicht mit angegeben, weil es nur die Betriebsart „Automatikbetrieb“ gibt. Im späteren Beispiel mit Einrichtbetrieb (Abschnitt 6.11) sind die Sicherheitsfunktionen einschließlich mehrerer Betriebsarten dargestellt.

## 6.4 Spezifikation der Steuerungshardware

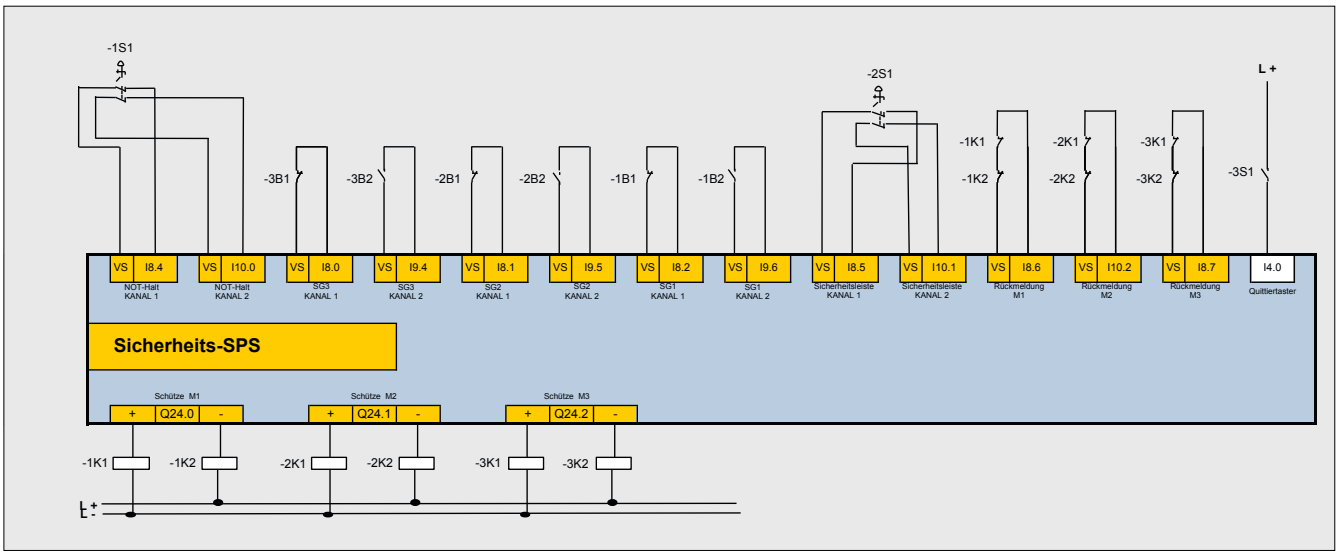
Abbildung 12 zeigt den Stromlaufplan (Dokument A2.2) des Beispiels. Er ist sehr einfach dargestellt. Üblicherweise liegen für dieses Dokument Schaltpläne aus einem Elektro-CAE-Programm vor.

Dem Stromlaufplan ist zu entnehmen, dass alle Sicherheitsfunktionen zweikanalig ausgeführt sind, aufgrund des erforderlichen Performance Levels $PL_r$ d. Auch die Antriebe werden über zwei in Reihe geschaltete Schütze geschaltet. Diese Schütze beaufschlagen allerdings jeweils nur einen Binärausgang. Dadurch geht die angestrebte Zweikanaligkeit scheinbar verloren. Jeder

Binärausgang wird jedoch steuerungsintern über zwei Schalter geschaltet (nicht dargestellt), sodass die abgebildete Lösung zulässig ist.

Abbildung 13 stellt den Systemaufbau dar. Dies ist konkret der Hardwareaufbau der Sicherheits-SPS und ggf. die Anbindung weiterer sicherer Komponenten.

Abbildung 12:
Stromlaufplan des Beispiels (Dokument A2.2)



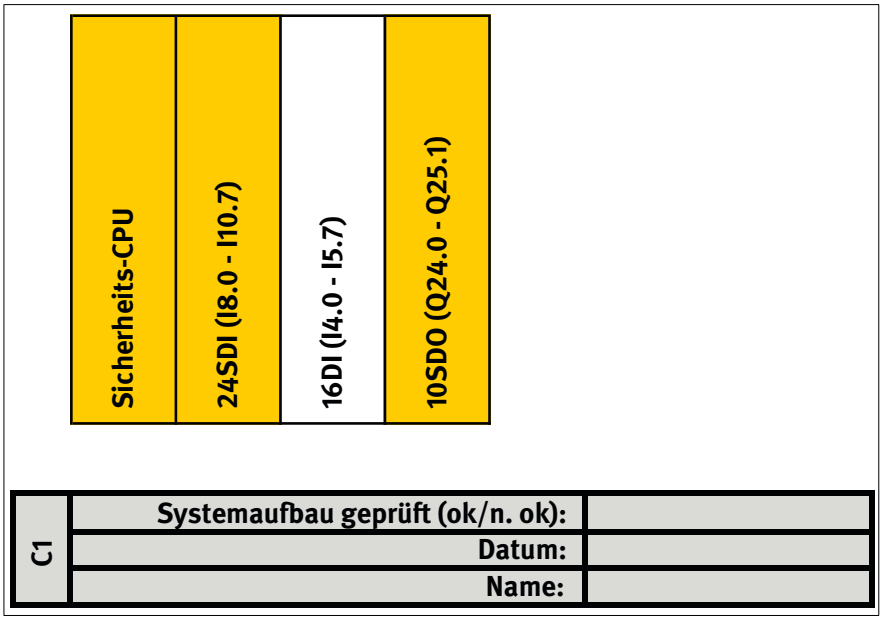




Abbildung 13:
Systemaufbau (Dokument A2.3)

In Gelb dargestellt sind die Sicherheits-SPS und sichere Ein- und Ausgangskarten. Die Standard-Eingangskarte ist nicht unterlegt. Grau unterlegt ist ein Prüffeld für das Codereview (C1).

Tabelle 8 zeigt die I/O-Liste (Dokument A2.4). Variablen beginnend mit der hier beispielhaft gewählten Syntax „IS_...“ oder „QS_...“ stammen aus dem sicheren I/O-Bereich der Sicherheits-SPS. Bei Datenaustausch über Netzwerke können hier anstelle der I/O-Adressen auch Kommunikationsparameter eingetragen werden.

Die I/O-Liste enthält zusätzliche Prüffelder (dunkelgrauer Bereich), die Teil des Validierungsprotokolls D1 und des Codereviews C1 sind. Dort wird die korrekte Verdrahtung der Sensoren und Aktoren (Spalte D1) und auch die korrekte Verschaltung der Variablen mit den Funktionsbausteinen in der Software (Spalte C1) überprüft. Die Validierung mit der Spalte D1 erfolgt an der aufgebauten Maschine, die Verifikation mit den Spalten C1 anhand des Programmlistings bzw. im Programmeditor nach der Codierung.

Bei zweikanaligen Eingängen, die durch eine I/O-Karte auf Diskrepanz getestet werden, ist nur die richtige Verdrahtung der Eingänge zu testen. Für die vorgegebene Diskrepanzzeit (z. B. 50 ms) muss die richtige Parametrierung auf der I/O-Karte überprüft werden. Dies zählt zu den herstellerspezifischen Tests.

Tabelle 8:
I/O-Liste (Dokument A2.4); NC = Öffner (normally closed), NO = Schließer (normally open)

| Signale | Variable | Adresse | D1<br>Validiert<br>(ok/n. ok) | C1<br>richtige Verschaltung in der Software verifiziert<br>(ok/n. ok) |
|---|---|---|---|---|
| **Eingänge** | | | | |
| Not-Halt EMST, zweikanalig (NC) (1S1) | IS_EMST | %I8.4 | | |
| Kontakt 1 Schutztür Roboter SG1 (NC) (1B1) | IS_SG1_1 | %I8.2 | | |
| Kontakt 2 Schutztür Roboter SG1 (NO) (1B2) | IS_SG1_2 | %I9.6 | | |
| Kontakt 1 Schnelllauftor SG2 (NC) (2B1) | IS_SG2_1 | %I8.1 | | |
| Kontakt 2 Schnelllauftor SG2 (NO) (2B2) | IS_SG2_2 | %I9.5 | | |
| Kontakt 1 Hubtor SG3 (NC) (3B1) | IS_SG3_1 | %I8.0 | | |
| Kontakt 2 Hubtor SG3 (NO) (3B2) | IS_SG3_2 | %I9.4 | | |
| Sicherheitsleiste von SG2, zweikanalig (NC) (2S1) | IS_SL_SG2 | %I8.5 | | |
| Rückmeldung Schütze M1 (NC) (1K1, 1K2) | IS_SM1 | %I8.6 | | |
| Rückmeldung Schütze M2 (NC) (2K1, 2K2) | IS_SM2 | %I10.2 | | |
| Rückmeldung Schütze M3 (NC) (3K1, 3K2) | IS_SM3 | %I8.7 | | |
| Quittiertaster (NO) (3S1) | I_ACK | %I4.0 | | |
| **Ausgänge** | | | | |
| Schütze Motor M1 (1K1, 1K2) | QS_M1 | %Q24.0 | | |
| Schütze Motor M2 (2K1, 2K2) | QS_M2 | %Q 24.1 | | |
| Schütze Motor M3 (3K1, 3K2) | QS_M3 | %Q 24.2 | | |
| | Datum: | | | |
| | Name: | | | |
| | Softwaresignatur: | | | |

Folgende Anmerkungen gibt es noch zum Stromlaufplan und zur I/O-Liste:

- Die zweikanaligen Elemente Not-Halt-Taster (IS_EMST) und Sicherheitsleiste (IS_SL_SG2) besitzen nur eine Adresse, da die beiden Kanäle bereits auf der I/O-Karte auf Diskrepanz überwacht werden. Die Karte gibt dann nur ein logisch komprimiertes Signal (siehe oben) weiter. Dies hat den Vorteil, dass das Programm die Diskrepanzüberwachung nicht mehr durchführen muss.

- Für die Schutztürkontakte ist eine Diskrepanzüberwachung auf der I/O-Karte nicht notwendig, da der verarbeitende Softwarebaustein SF_GuardMonitoring die beiden Kontaktinformationen separat benötigt.

- Die Spiegelkontakte eines Schützpaares werden in Reihe geschaltet und jeweils in einen Binäreingang eingelesen (IS_SM1, 2, 3). Damit lässt sich mit dem Funktionsbaustein SF_EDM der Ausfall des Schützes überwachen.

- Wird durch die Schütze nur die Steuerspannungsversorgung der eigentlichen Leistungsschütze eines Motors geschaltet, so müssen für die Schützüberwachung die Spiegelkontakte aller beteiligten Schütze (Steuer- und Leistungsschütze) in Reihe geschaltet und eingelesen werden.

## 6.5 Katalog der fehlervermeidenden Maßnahmen

Tabelle 9 zeigt beispielhaft den Katalog der übergeordneten fehlervermeidenden Maßnahmen und Tabelle 10 die für die verwendete Steuerung spezifischen fehlervermeidenden Maßnahmen (beides zusammengestellt in Dokument A3). Weitere Informationen zu den Maßnahmen finden sich in Abschnitt 5.7.

Die hier dargestellten Maßnahmen sind als Beispiele zu verstehen. Dieser Katalog ist entsprechend den Anforderungen des Unternehmens und des Projektes für eine konkrete Anwendungsprogrammierung und für die verwendete Steuerung geeignet anzupassen und zu ergänzen.

Die festgelegten fehlervermeidenden Maßnahmen sind in Dokument A3 (wie in Tabelle 9 und Tabelle 10) durchnummeriert (Rx) und können in der Spalte C1 beim Codereview als umgesetzt („ok“) oder nicht umgesetzt („n. ok“ für nicht ok) gekennzeichnet werden.

Tabelle 9:
Beispielkatalog der übergeordneten fehlervermeidenden Maßnahmen (Dokument A3)

| | C1 | |
|---|---|---|
| | Kürzel | ok/n. ok |
| **Variablen** | | |
| **Präfix für Binäreingänge:** I_... (nicht sicherheitsrelevant) / IS_... (sicherheitsrelevant) | R1 | |
| **Präfix für Binärausgänge:** Q_... (nicht sicherheitsrelevant) / QS_... (sicherheitsrelevant) | R2 | |
| **Präfix für Instanzen:** Timer: „T_“, Positive Flankenerkennung: „R_“, Flip-Flops: „FF_“, SF_Guardmonitoring: „MON_“, SF_EmergencyStop: „EMST_“, SF_EDM: „EDM_“ | R3 | |
| **Präfix von globalen Variablen:** „G_“ (nicht sicherheitsrelevant)/ „GS_“ (sicherheitsrelevant). | R4 | |
| **Variablennamen:** Der Variablenname nach dem Präfix sollte selbsterklärend sein, z. B. mit der Bezeichnung der technischen Einrichtung (...SG1... für SG1). | R5 | |
| **Variablendeklaration:** Mit sicherem Zustand initialisieren. Kommentar notwendig. | R6 | |
| **Kommentare:** Jedes Netzwerk und jede Variablendeklaration erhält einen Kommentar. | R7 | |
| **Signalverarbeitung** | | |
| **Softwarearchitektur:** Die Software sollte in eine Vorverarbeitungsebene, eine Abschaltlogik und eine Nachverarbeitungsebene strukturiert werden. Die Vorverarbeitungsebene sollte in nacheinander folgenden Netzwerken realisiert werden. Jeder Binärausgang sollte zusammen mit der Abschaltlogik und der Nachverarbeitungsebene in einem Netzwerk realisiert werden. | R8 | |
| **Zuweisung:** Variablen sollten nur an einer Stelle zugewiesen werden. | R9 | |
| **Zyklische Abarbeitung:** Jeder Teil der Software wird ohne Bedingungen zyklisch abgearbeitet. | R10 | |
| **Überwachung von zweikanaligen Tastern:** Zweikanalige Taster werden auf der Eingangskarte mit einer Diskrepanzzeit von 100 ms überwacht. | R11 | |
| **Überwachung von Schützen:** Schütze werden mit einer Diskrepanzzeit von 1 s überwacht. | R12 | |
| **Überwachung von Schutztüren:** Schutztürkontakte werden mit einer Überwachungszeit von 1s überwacht. | R13 | |
| **Automatischer Wiederanlauf:** Ist nur für Automatiktüren erlaubt. | R14 | |
| **Peripheriefehler:** Eine Quittierung ist notwendig. | R15 | |
| **Aktivierung von Sicherheitsfunktionen:** Sicherheitsfunktionen werden durch ein FALSE-Signal aktiviert. | R16 | |
| **Benutzte Bibliotheksbausteine** | | |
| **Verwendung:** Bibliotheksbausteine sollen vorzugsweise verwendet werden. | R17 | |
| **Schutztüren:** SF_GuardMonitoring | R18 | |
| **Not Halt:** SF_EmergencyStop | R19 | |
| **Schütze:** SF_EDM | R20 | |
| **Automatischer Wiederanlauf:** Die Parameter „S_StartReset“ und „S_AutoReset“ der Bibliotheksbausteine sind FALSE. Nur für Automatiktüren darf S_AutoReset = TRUE gesetzt werden. | R21 | |
| **Aktivierung:** Der Eingangsparameter „Activate“ von SF_GuardMonitoring, SF_EmergencyStop und SF_EDM ist immer TRUE. | R22 | |
| **Selbst entwickelte Module:** Wenn möglich, sollten logische Verknüpfungen, die mehrfach Verwendung finden, in einem Modul als Funktion oder Funktionsbaustein gekapselt werden. Die Entwicklung erfolgt nach dem V-Modell. Ein Passwortschutz und ein Bibliotheksmanagement sind notwendig. | R23 | |
| **Aktivitäten nach Änderungen** | | |
| **Dokumentation:** Alle Änderungen müssen in der Änderungshistorie dokumentiert werden. | R24 | |
| **Validierung:** Nach Änderungen muss die Validierung der geänderten Software wiederholt werden. | R25 | |
| | Datum: | |
| | Name: | |

Tabelle 10:
Beispielkatalog der steuerungsspezifischen fehlervermeidenden Maßnahmen (Dokument A3)

| | | C1 | |
|---|---|---|---|
| | | Kürzel | ok/n. ok |
| **Programmeditor/Programmiersprache** | | | |
| Genutzter Programmeditor | Safety Editor V10.1 | R26 | |
| Programmiersprache | Funktionsbausteinsprache (FBD) | R27 | |
| Softwarebibliothek | Safety Library V3.2 | R28 | |
| **Signalverarbeitung** | | | |
| Die sicheren I/O-Karten müssen mit ACK_XYZ betrieben werden, d. h. nach Beseitigung eines Fehlers findet keine automatische Wiedereingliederung der I/O-Karten statt. | | R29 | |
| Nach einer Passivierung der I/O-Karten muss die Wiedereingliederung mit Quittierung erfolgen. | | R30 | |
| Sicherheitsbetrieb muss in der SPS aktiviert sein (kein Testbetrieb). | | R31 | |
| Wenn Testbetrieb aktiv ist, sollte das angezeigt werden und Maschine automatisch abschalten. | | R32 | |
| | | Datum: | |
| | | Name: | |

## 6.6 Architektur des Sicherheits-programms und des Standard-programms

Abbildung 14 zeigt die Architektur des Sicherheitsprogramms (Dokument B1). Die Aufrufhierarchie kann je nach Steuerungssystem unterschiedlich aussehen. Zu erkennen ist allerdings, dass das Sicherheits-Hauptprogramm (FB_Main) die Sicherheits-Bibliotheksbausteine aufruft. Im Bild enthalten sind auch eine kurze Erklärung der Bedeutung der einzelnen Bausteine sowie ein Prüffeld für das Codereview (C1).

Die Architektur des Standardprogramms (Dokument B2) ist optional, da es üblicherweise für die Abarbeitung des Sicherheitsprogramms nicht relevant ist, für das Verständnis der Gesamtfunktion der Anlage aber hilfreich sein kann.

Bei einfachen Anwendungen kann die Architektur des Sicherheitsprogramms (Dokument B1) entfallen.

In Abbildung 15 ist die Modularchitektur (Dokument B3) zu sehen. Diese zeigt das Zusammenspiel der gesamten Sicherheitssoftware. Man kann der Abbildung eine klare Dreiteilung entnehmen. Auf der linken Seite befindet sich die Vorverarbeitungsebene, in der die Eingänge vorwiegend mit Bibliotheksbausteinen verarbeitet werden. Auf der rechten Seite ist die Ansteuerungsebene dargestellt, in der Ausgänge mit Bibliotheksbausteinen oder direkt angesteuert werden. Diese beiden Teile sind vom Aufbau her durch die Peripherie vorgegeben. In der Mitte ist die noch zu spezifizierende Ansteuerlogik der Aktoren blau dargestellt (Modul ACT, von englisch: to actuate = ansteuern). Darunter befindet sich ein Feld für die Verifikation (V1) der Modularchitektur. Verifiziert wird gegen die Spezifikation der Sicherheitsfunktionen (Dokument A1). Die Modularchitektur kann anstelle der grafischen Darstellung wie in Abbildung 15 auch in Tabellen als Liste der Module dargestellt werden. Der Beschreibungsaufwand verringert sich dadurch. Das Tool SOFTEMA (Kapitel 14) verwendet die Listenform.

Die in Abbildung 15 dargestellten zertifizierten Funktionsbausteine nach PLCopen [8] SF_EmergencyStop (Not-Halt-Baustein), SF_GuardMonitoring (Schutztür-Baustein) und SF_EDM (Schützüberwachungs-Baustein) sind in Abschnitt 6.17 in ihrer wesentlichen Funktionalität erläutert. Bei den Funktionsbausteinen sind nicht alle Signale dargestellt, sondern nur die für den weiteren Verlauf wichtigen Signale. Die Ausgänge der Vorverarbeitungsebene sind in negativer Logik (bzw. low-aktiv, active low).

Derjenige Teil der Abbildung 15, der noch unbekannt ist, ist das Ansteuerungsmodul ACT der Aktoren. ACT besteht im allgemeinen Fall nur aus den elementaren Grundverknüpfungen UND (&), ODER (≥1) und NICHT (o). Die Symbole für die hier verwendeten logischen Operatoren entsprechen IEC 60617-12. Die grundsätzliche Struktur des Moduls ACT für die Ansteuerung eines Aktors zeigt Abbildung 16.

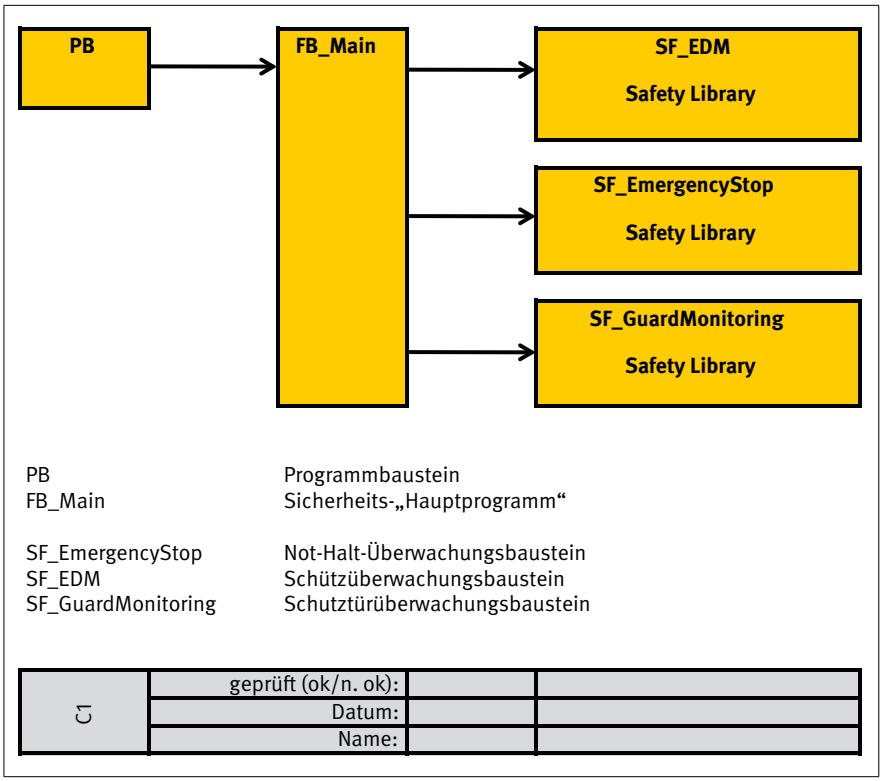
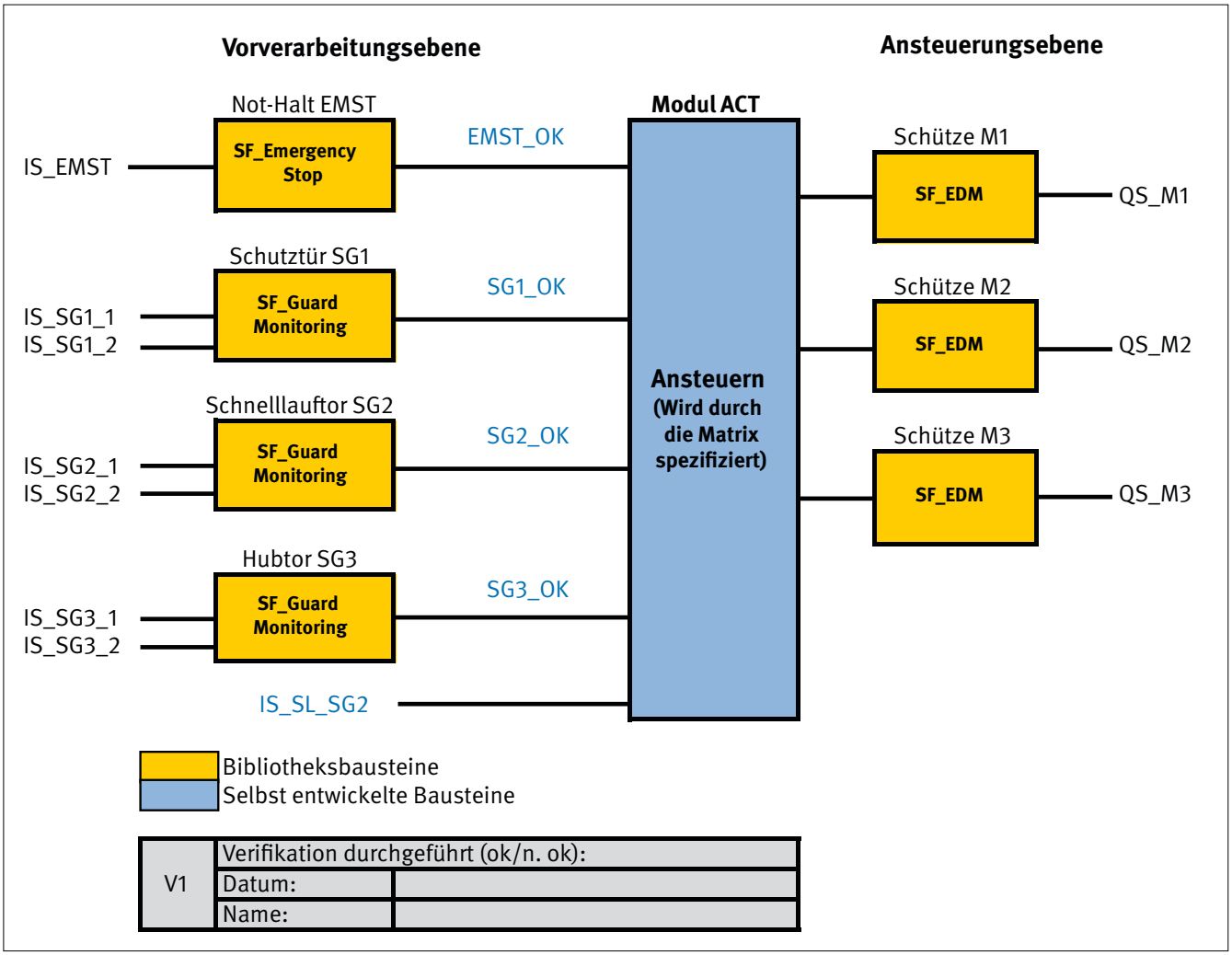



Abbildung 14:
Architektur des Sicherheitsprogramms (Dokument B1), hier als Aufrufhierarchie

Abbildung 15:
Modularchitektur (Dokument B3)



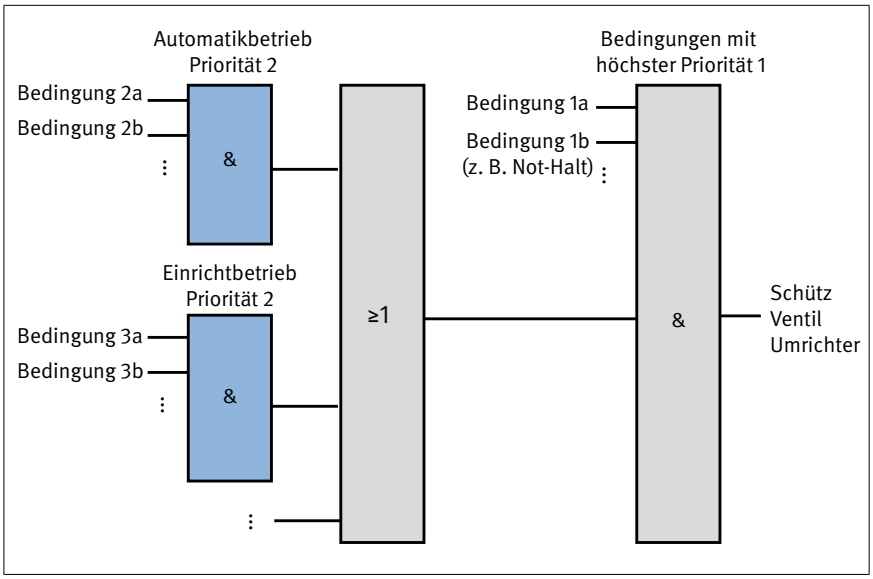



Abbildung 16:
Struktur des Moduls ACT für einen Aktor

Sicherheitsfunktionen werden – dem Ruhestromprinzip entsprechend – im Allgemeinen mit dem 0-Signal aktiviert, d. h. die Gefahr wird abgestellt. Wenn der Ausgang des rechten UND-Gliedes FALSE ist, wird ein Schütz abgeschaltet oder ein Ventil geht in seine Sicherheitsstellung oder eine im Umrichter integrierte Sicherheitsfunktion wird aktiviert. Abbildung 16 zeigt auch die Bedeutung der Prioritäten (1 ist hier die höchste Priorität). Prioritäten sind meistens mit Betriebsarten verknüpft. Signale mit der Priorität 1 aktivieren mit einem 0-Signal die Sicherheitsfunktion auf jeden Fall. Darunter kann man sich z. B. den Not-Halt vorstellen. Signale, die z. B. dem Automatikbetrieb bzw. dem Einrichtbetrieb zugeordnet sind, besitzen die gleiche Priorität 2.

Sollten in seltenen Fällen sogar Signale der Priorität 3 notwendig sein, so würden diese mit der gleichen ODER-UND-Kombination als Eingang an eines der blauen UND-Glieder der Priorität 2 verschaltet.

Im obigen Beispiel gibt es jedoch nur eine Betriebsart, sodass hier nur das rechte UND-Glied pro Aktor zum Tragen kommt und somit die Prioritäten 1 und 2 gleichbedeutend sind.

## 6.7 Softwarespezifikation mit der Cause-and-Effect-Matrix

Die oben beschriebene Logik des Ansteuermoduls ACT muss im nächsten Schritt für die Codierung spezifiziert werden. Diese Spezifikation soll zusätzlich für die Verifikation und Validierung des Programms verwendbar sein. Dazu wird eine Cause-and-Effect (C&E)-Matrix aufgestellt. Es handelt sich um eine grafische Darstellung von Ursachen (causes), die zu Ergebnissen (effects) führen oder diese maßgeblich beeinflussen. Für diese Darstellungsform finden sich im Englischen mehrere synonyme Bezeichnungen: „cause and effect diagramm“, „cause and effect charts“, „cause and effect tables“ oder die hier verwendete Bezeichnung „cause and effect matrix“. Im Deutschen heißt es auch Ursache-Wirkungs-Diagramm. Für die hier vorgestellte Methode wird die Darstellung als „C&E-Matrix“ bezeichnet. In der Norm IEC 62881 werden diese Darstellungsformen beschrieben. Ein Beispiel für die C&E-Matrix der betrachteten Anlage zeigt Abbildung 17.

Abbildung 17:
Cause-and-Effect(C&E)-Matrix

| Cause | | | | | | | | | Effect | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Involvierte Eingänge | | | | | | | | Sicherheitsfunktionen | Ausgänge | | |
| IS_EMST (I8.4) | IS_SG1_1 (I8.2) | IS_SG1_2 (I9.6) | IS_SG2_1 (I8.1) | IS_SG2_2 (I9.5) | IS_SG3_1 (I8.0) | IS_SG3_2 (I9.4) | IS_SL_SG2 (I8.5) | | QS_M1 (Q24.0) | QS_M2 (Q24.1) | QS_M3 (Q24.2) |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ALL_OK | EIN | EIN | EIN |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | SF1: Not-Halt EMST betätigt | AUS | AUS | AUS |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | SF2: SG1 offen | AUS | NOP | NOP |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | SF3: SG2 offen | NOP | AUS | NOP |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | SF4: SG2 und SG3 offen | AUS | NOP | NOP |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | SF5: Sicherheitsleiste betätigt | NOP | NOP | AUS |

Die weiteren Erörterungen beziehen sich darauf. Bei einer C&E-Matrix stehen auf der linken Seite in Zeilen untereinander die auslösenden Ereignisse (Causes, das sind die Signaleingänge der Steuerung) und rechts spaltenweise die Auswirkungen (Effects, das sind die Signalausgänge). Für die Beispielanlage gibt es einen Startzustand, von dem aus sämtliche Sicherheitsfunktionen getestet werden. Dieser Zustand wird in der Matrix als „ALL_OK“ bezeichnet. Im Zustand ALL_OK sind alle Freigabesignale für die Aktoren QS_M1 ... QS_M3 TRUE. Von diesem Startzustand aus werden die Schaltvorgänge der Sicherheitsfunktionen mit den zugehörigen Eingangssignalen eingetragen.

Diese Matrix zeigt eindeutig das Schaltverhalten der Sicherheitsfunktionen. Damit dient sie als Testgrundlage für die Softwarevalidierung. Diese Matrix wird schon in der Spezifikationsphase, unabhängig von der späteren Programmierung, aufgestellt.

Da die Matrix das Schaltverhalten der Sicherheitsfunktionen eindeutig zeigt, ist es naheliegend, diese zur Spezifikation der Software der Sicherheitsfunktionen zu erweitern. Dazu wird die einfache Darstellung aus Abbildung 17 ergänzt. Weiterhin kommen Felder für Codereview und Validierung hinzu sowie Eintragungen, die das Ansteuermodul ACT eindeutig spezifizieren. Daraus ergibt sich dann das Dokument B4 „Sicherheitsbezogene Softwarespezifikation und Validierungsplan“. Abbildung 18 zeigt die vollständige Matrix.

Abbildung 18:
Erweiterte C&E-Matrix (Dokument B4 „Sicherheitsbezogene Softwarespezifikation und Validierungsplan“); Prüffelder (graue Felder) und Steuerungsinformationen (blaue Variablen)

| | | | Cause | | | | | | | | | Effect | | | | D1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Involvierte Eingänge | | | | | | | | Sicherheitsfunktionen | Ausgänge | | | | | | |
| Betriebsart | Vorgängerzustand bei Test | Zustand | IS_EMST (I8.4) | IS_SG1_1 (I8.2) | IS_SG1_2 (I9.6) | IS_SG2_1 (I8.1) | IS_SG2_2 (I9.5) | IS_SG3_1 (I8.0) | IS_SG3_2 (I9.4) | IS_SL_SG2 (I8.5) | | QS_M1 (Q24.0) | QS_M2 (Q24.1) | QS_M3 (Q24.2) | Quittierung I_ACK (I4.0) | Geprüft (ok/n.ok) | Name | Datum |
| | | | | | | | | | | | C1: Software entspricht der Matrixdokumentation | | | | | | | |
| | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ALL_OK | EIN | EIN | EIN | | | | |
| | 1 | 2 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | SF1: Not-Halt betätigt | EMST_OK AUS | EMST_OK AUS | EMST_OK AUS | EIN | | | |
| | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | SF2: SG1 offen | SG1_OK AUS | NOP | NOP | EIN | | | |
| | 1 | 4 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | SF3: SG2 offen | NOP | SG2_OK AUS | NOP | EIN | | | |
| | 1 | 5 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | SF4: SG2 und SG3 offen | SG2_OK v SG3_OK AUS | NOP | NOP | EIN | | | |
| | 1 | 6 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | SF5: Sicherheitsleiste betätigt | NOP | NOP | IS_SL_SG2 AUS | EIN | | | |
| V1 | Verifikation durchgeführt (ok/n. ok): | | | | | | | | | | | | | Softwaresignatur: | | | | |
| | Datum: | | | | | | | | | | | | | | | | | |
| | Name: | | | | | | | | | | | | | | | | | |

In den zusätzlichen Spalten am linken Rand werden für jede Sicherheitsfunktion die Betriebsart, eine laufende Nummer (Zustand) sowie der Vorgängerzustand beim Testen der Sicherheitsfunktion angegeben. Der Vorgängerzustand bedeutet: Für den Test wird dieser Zustand eingestellt und dann die zu testende Sicherheitsfunktion angefordert.

Die in Abbildung 18 unten aufgeführte „Softwaresignatur“ ist ein eindeutiges Kennzeichen für eine Version der SRASW. Wird das Programm auch nur an einer Stelle geändert, so ergibt sich eine andere Softwaresignatur.

Das Feld „Verifikation“ (V1) links unten in Abbildung 18 dient dazu, die Schaltinformationen (AUS, EIN, NOP) und die blau geschriebenen Variablen gegenüber der Spezifikation der Sicherheitsfunktionen (Dokument A1) zu verifizieren.

Die Quittierung zeigt sowohl den Quittiereingang als auch die Notwendigkeit einer Quittierung an. Ist beispielsweise die Sicherheitsfunktion SF1 ausgelöst worden, kommt man erst nach Entriegelung des NOT-HALT-Tasters und dann einer Quittierung mit „EIN“ (entspricht TRUE des Quittiereingangs) wieder in den Zustand ALL_OK.

In Abbildung 18 sind in blauer Schriftfarbe diejenigen Variablen eingetragen, die relevant für die Softwarespezifikation sind. Die Variablen sind die Ausgangsgrößen aus der Vorsteuerungsebene und damit Eingangsgrößen des Ansteuermoduls ACT (Abbildung 15). Diese Variablen sind in negativer Logik.

Das Bildungsgesetz für die Softwarespezifikation lautet:

Schritt 1: Für jede einzelne Sicherheitsfunktion, die bei einem Aktor einen Schaltvorgang auslöst (bezogen auf den Vorgängerzustand, hier: ALL_OK), trägt man in die entsprechende Zelle der Tabelle diejenige logische Verknüpfung der Eingangsgrößen von ACT ein, die den Schaltvorgang auslöst. Dieser Schaltvorgang wird hier durch „AUS“ bzw. „EIN“ angegeben. Löst eine Sicherheitsfunktion bei einem Aktor keinen Schaltvorgang aus, ist dort ein „NOP“ einzutragen.

Ein Beispiel aus Abbildung 18 für die Sicherheitsfunktion SF1 und wie sie auf den Ausgang QS_M3 wirkt:

In der Zelle steht „EMST_OK“ und darunter „AUS“. Dieser Eintrag ist so zu lesen: „Wenn die Variable EMST_OK = FALSE ist, dann soll Ausgang QS_M3 = FALSE sein.“

NOP löst keinen Schaltvorgang aus.

Hier ein Beispiel für die Sicherheitsfunktion SF4 und den Ausgang QS_M1:

SF4 löst einen Schaltvorgang genau dann aus, wenn die Schutztüren SG2 und SG3 offen sind. Deshalb muss die ODER-Verknüpfung „SG2 v SG3“ eingetragen werden. Das liest sich: „Wenn der Ausdruck (SG2 _OK ODER SG3_OK) = FALSE ist, dann soll Ausgang QS_M1 = FALSE sein.“

Schritt 2: Die komplette logische Verknüpfung pro Aktor (also über alle Sicherheitsfunktionen) ergibt sich dann aus der UND-Verknüpfung der in der Spalte des Aktors stehenden Variablen in blauer Schriftfarbe.

Abbildung 19 zeigt das Endergebnis für ACT in diesem Beispiel.

Es wäre möglich, diesen Code aus den blau geschriebenen Eintragungen in Abbildung 18 automatisch zu generieren, wenn diese Möglichkeit in der Programmierumgebung einer Sicherheits-SPS angeboten würde. Bestünde diese Möglichkeit, so könnte in der Programmierumgebung der Sicherheits-SPS mit der Spezifikation des Moduls ACT auch dessen Programmierung im gleichen Arbeitsvorgang erledigt werden.

Für die Anwendung der Matrix gilt:

- Programmierende lesen pro Aktor die zugehörige Spalte der C&E-Matrix bzgl. der blau geschriebenen Eintragungen, um daraus die Logik zu codieren.
- Testende lesen die Zeilen der Matrix, um einzelne Sicherheitsfunktionen zu testen.

Durch die C&E-Matrix wird das gesamte Schaltverhalten der Sicherheitsfunktionen transparent und verifizierbar dargestellt. Durch die blau geschriebenen Eintragungen wird die Software des Moduls ACT eindeutig formal spezifiziert. Weiterhin wird die Software dadurch im Codereview übersichtlich, nachvollziehbar und verifizierbar.

Für das Beispiel wird in Abbildung 20 die komplette Sicherheits-Programmskizze (Dokument B5) dargestellt. Diese Skizze dient zum Verständnis des Beispiels und wird in der praktischen Anwendung der Matrixmethode eher keine Rolle spielen, weil stattdessen auf der Basis der Spezifikation B4 direkt das Programm codiert wird.

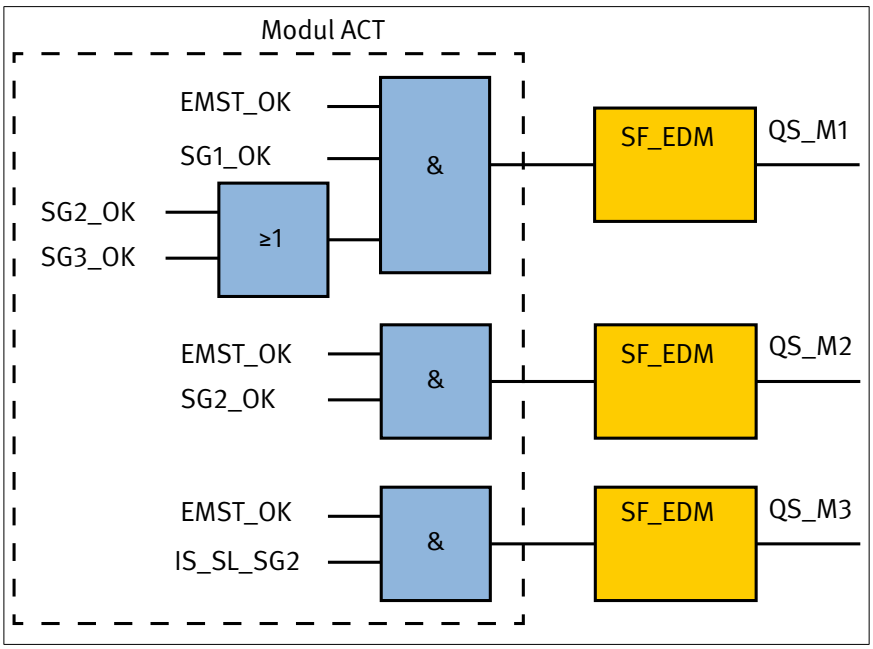
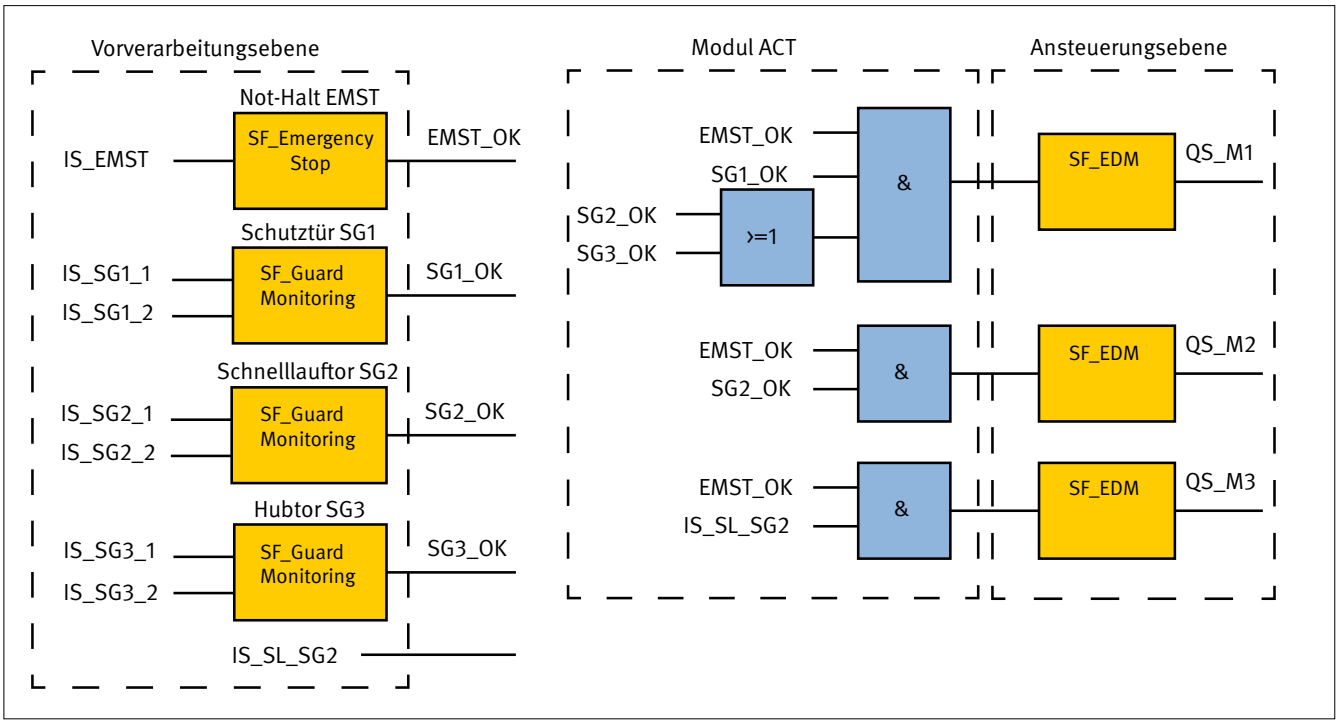



Abbildung 19:
Logischer Aufbau der Ansteuerung ACT (im gestrichelten Rahmen)

Abbildung 20:
Programmskizze (Dokument B5)




## 6.8 Verifikation und Validierung in der Matrixmethode des IFA

Die C&E-Matrix in Abbildung 18 enthält weiterhin die folgenden Felder:

- V1 Verifikation:
  Entspricht das eingetragene Schaltverhalten den in A1 spezifizierten Sicherheitsfunktionen?

- C1 Codereview:
  Entspricht der Programmcode des Ansteuermoduls ACT dem eingetragenen Schaltverhalten (spaltenweise je Ausgang vorgehen)?

- D1 Softwarevalidierung:
  Entspricht die Anwendungssoftware den Sicherheitsanforderungen (zeilenweise je Sicherheitsfunktion vorgehen)? Sämtliche Sicherheitsfunktionen werden in der Anlage real getestet. Die zugehörigen Zustände und Adressen der Ein- und Ausgänge sind ebenfalls aufgeführt. Weiterhin ist dargelegt, von welchem Ausgangszustand aus jeweils zu testen ist. Man beachte, dass zusätzliche Zeilen zum Test mit weiteren Testfällen problemlos in die Matrix eingefügt werden können. Ein Vorabtest per Simulation ist entsprechend den normativen fehlervermeidenden Maßnahmen bei $PL_r$ c, d und e empfohlen.

Tabelle 11 zeigt ein Beispiel für ein Codereview (Dokument C1). Das Codereview verweist in der Spalte „Referenz“ auf relevante Dokumente. In diesen Dokumenten ist jeweils ein Feld, eine Spalte oder eine Zeile „C1“ für das Codereview vorgesehen.

Tabelle 11:
Beispiel für ein Codereview (Dokument C1)

| Wurden die Aktivitäten durchgeführt? | Referenz | durchgeführt (ja/nein) |
|---|---|---|
| 1. Sind die vereinbarten fehlervermeidenden Maßnahmen, Tools und Programmierregeln bei der Codierung eingehalten? | A3 Maßnahmen (Spalte C1) | |
| 2. Ist der Systemaufbau der Hardware umgesetzt? | A2.3 Systemaufbau (Feld C1) | |
| 3. Ist die Verschaltung der I/O-Signale im Code korrekt umgesetzt? | A2.4 IO-Liste (Spalte C1) | |
| 4. Ist die Architektur des Sicherheitsprogramms eingehalten? | B1 Architektur Sicherheitsprogramm (Feld C1) | |
| 5. Ist die Modularchitektur eingehalten? | B3 Modularchitektur (Feld C1) | |
| 6. Ist die Spezifikation aus der Matrix im Code umgesetzt? | B4 C&E-Matrix (Zeile C1) | |
| Datum: | | |
| Name: | | |
| Softwaresignatur: | | |

Das Protokoll der Softwarevalidierung (Dokument D1, Beispiel siehe Tabelle 12) verweist auf die in den Referenzdokumenten mit „V1“ bzw. „D1“ gekennzeichneten Felder, Spalten oder Zeilen.

Tabelle 12:
Beispiel für ein Protokoll der Softwarevalidierung (Dokument D1)

| Wurden die Aktivitäten durchgeführt? | Referenz | durchgeführt (ja/nein) |
|---|---|---|
| Verifikation der Modularchitektur | B3 Modularchitektur (Feld V1) | |
| Verifikation der Matrix | B4 C&E-Matrix (Feld V1) | |
| Codereview | C1 Codereview | |
| I/O-Check | A2.4 I/O-Liste (Spalte D1) | |
| Prüfung der Peripheriegeräte | siehe Herstellervorgaben | |
| Validierung C&E-Matrix | B4 C&E-Matrix (Spalte D1) | |
| Validierung Sicherheitsfunktionen | A1 Sicherheitsfunktionen (Spalte D1) | |
| Validierung Anforderungen | A4 Anforderungen (Spalte D1) | |

| Ist die Dokumentation komplett? | vorhanden (ja/nein) |
|---|---|
| Dokumente des V-Modells aus diesem Excel-Dokument | |
| PDF-Ausdruck aller sicherheitsrelevanten Software inkl. Checksumme | |
| PDF-Ausdruck der Hardwarekonfiguration (mit allen Einstellungen) inkl. Checksumme | |
| Archivierung der PDF-Handbücher aller Systemkomponenten | |
| PDF-Ausdruck der Konfiguration von Peripheriegeräten inkl. Checksummen | |
| Abnahmevorschriften der Hersteller (z.B. Parametrierung von Sicherheitskomponenten) | |
| Relevante Normen | |
| Datum: | |
| Name: | |
| Softwaresignatur: | |

Für den Anwender stellt sich hier die Frage nach der Testabdeckung der Spalten „D1“ in Abbildung 18. Hier wird nur die Funktion der Sicherheitsfunktionen (Anzahl 5) getestet, wobei – wie bereits erwähnt – eventuell Zeilen mit zusätzlichen Testfällen hinzugefügt werden können.

Betrachtet man Abbildung 15, so erkennt man, dass das Modul ACT hier fünf binäre Eingangsgrößen (d. h. die Freigabesignale) und drei binäre Ausgangsgrößen (Aktorsignale) besitzt. Das Schaltverhalten von ACT ließe sich also durch eine Schalttabelle mit 32 Zeilen (aus der Kombinatorik: $2^5$ Eingangsgrößen) und drei Spalten beschreiben. Durch den Einsatz von mathematischen Umformungen würde man ebenfalls die Logik in Abbildung 19 erhalten. Die vollständige Testabdeckung bzgl. des Moduls ACT wäre die Abarbeitung dieser Schaltmatrix. Dieser Aufwand wäre hier in der Praxis gerade noch vertretbar.

Anders sieht es aus, wenn z. B. zehn Sicherheitsfunktionen vorhanden sind, sodass ACT näherungsweise ca. zehn Eingangsgrößen besitzt. Zehn und mehr Sicherheitsfunktionen können in realen Anwendungen ohne Weiteres vorkommen. Dann besteht die Schaltmatrix aus 1 024 Zeilen und entsprechenden Spalten für die Aktoren. Der Spezifikations- und Testaufwand mittels der Schaltmatrix wäre hier nicht vertretbar. Allein beim Ausfüllen der Schalttabelle werden sich wahrscheinlich einige Fehler einschleichen. Würde man eine ähnliche Matrix wie in Abbildung 18 benutzen, so wären zehn Zeilen notwendig, die überschaubar, inkl. der blau hinterlegten Eintragungen, auszufüllen sind. Hier wird dafür plädiert, in der Praxis bei der Softwarevalidierung einen pragmatischen Weg einzuschlagen, der zwar keine vollständige Testabdeckung liefert, aber von einem breiten Personenkreis nachvollziehbar durchgeführt werden kann. Dies ist z. B. mit den in Tabelle 8 und Abbildung 18 vorgeschlagenen Validierungsspalten „D1“ möglich. Damit wird auch vollkommen transparent, was, wann und von wem geprüft worden ist. Hinweise zu einer angemessenen Testabdeckung finden sich in Abschnitt 5.11.

Die Softwarevalidierung (Dokument D1) besteht aus den Teilen

- Analyse: der Verifikation (Dokumente B3, B4), des Codereviews (C1), der normativen Anforderungen (A4), der Sicherheitsfunktionen (A1) und
- realer Funktionstest: der I/O-Liste und der C&E-Matrix (Dokumente A2.4, B4).

Die Überprüfungsaktivitäten sollten vorzugsweise nach dem Vier-Augen-Prinzip durchgeführt werden (Abschnitt 5.15).

Zusätzlich zu den hier dargestellten Spezifikations- und Überprüfungsaktivitäten zur Anwendungssoftware müssen bei der Softwarevalidierung noch die Spezifikationen und Tests der herstellerspezifischen Parametrierungen der Systemkomponenten (Sicherheits-SPS, I/O-Karten, Umrichter, Sensoren usw.) erfolgen. Beispielsweise muss die korrekte sicherheitsbezogene

Parametrierung von Umrichtern überprüft und dokumentiert werden (siehe Dokumente in Tabelle 12).

Für die Validierung und auch für eine externe Prüfung der Software müssen alle nötigen Unterlagen erstellt und gesichert werden.

Im Einzelnen sollte die Dokumentation enthalten:

- Dokumente des V-Modells aus der Excel-Mappe,
- PDF-Ausdruck aller sicherheitsrelevanten Software inkl. Checksumme,
- PDF-Ausdruck der Hardwarekonfiguration (mit allen Einstellungen) inkl. Checksumme,
- Archivierung der Handbücher aller Systemkomponenten, da sich im Laufe der Zeit dort Änderungen ergeben können, sodass der verbaute Stand evtl. nicht mehr nachträglich dokumentierbar ist,
- PDF-Ausdruck der herstellerspezifischen Konfiguration von Peripheriegeräten inkl. Checksummen,
- Abnahmevorschriften der Hersteller (z. B. für die Parameter einer Sicherheits-SPS oder eines Sicherheitsumrichters); diese sind zumeist in den Systemhandbüchern enthalten,
- einzuhaltende Vorgaben aus C-Normen für Spezialmaschinen (z. B. Pressen),
- und weitere relevante Normen bzw. Dokumente.

Für die Dokumentation wird hier PDF als ein immer lesbares Datenformat vorgeschlagen, da externen Prüfenden nicht unbedingt jede Systemsoftware zur Verfügung steht.

## 6.9 Kompaktere Softwarespezifikation

Die Darstellung der C&E-Matrix in Abbildung 18 ist eventuell für größere Mengengerüste von Aktoren (z. B. mehrere Hundert Aktoren bei Walzwerken) und Sensoren nicht geeignet.

Deshalb sollen hier noch alternative, kompaktere Darstellungen beschrieben werden. Um größere Mengengerüste von Aktoren in einer Tabelle besser verwalten zu können, empfiehlt es sich, die Darstellung in Abbildung 18 zu transponieren. Mit SOFTEMA (Kapitel 14) wird diese Umwandlung automatisch durchgeführt. Allerdings steigt mit einem größeren Mengengerüst von Aktoren auch das Mengengerüst an relevanten Eingangsgrößen. Daher hätte man nach Transposition der C&E-Matrix sehr wahrscheinlich ein Darstellungsproblem bezüglich der Eingangsgrößen. Aus diesem Grund wird in dieser transponierten Darstellung die Information über die beteiligten Eingangsgrößen nur sehr rudimentär berücksichtigt. Damit ist die Annahme verbunden, dass die erfahrene testende Person bei der Softwarevalidierung weiß, was dann konkret bezüglich dieser Eingangsgrößen zu tun ist. Bei einer Schutztür muss z. B. bekannt sein, dass diese beim Test zu öffnen ist. Bei einem Not-Halt-Taster muss bekannt sein, dass dieser beim Test zu betätigen ist. Abbildung 21 zeigt die transponierte und reduzierte Darstellung der C&E-Matrix in diesem Beispiel.

Offensichtlich ist die Darstellung von Abbildung 21 wesentlich kompakter als die in Abbildung 18. Allerdings enthält sie auch weniger Detailinformationen, was aber in der industriellen Praxis kein Nachteil sein muss. Die Eintragungen in blauer Schrift spezifizieren jeweils komplett die Ansteuerungslogik des links stehenden Aktors. Darunter stehen summarisch die beteiligten Eingangsgrößen.

Bei dieser neuen Darstellung verlieren die Felder der Matrix, die die Abschaltungen beschreiben, allerdings den direkten Zusammenhang mit den in Tabelle 7 definierten Sicherheitsfunktionen, da diese sensororientiert formuliert sind. Die Spalte „beteiligte SFs“ soll hier der Übersicht dienen und den Zusammenhang zu den beteiligten Sicherheitsfunktionen herstellen.

Eine noch kompaktere Darstellung von Abbildung 21 besteht darin, die Eintragungen bzgl. der Eingangssignale wegzulassen (Abbildung 22), unter der Voraussetzung, dass erfahrene testende Personen anhand der Variablennamen erkennen, welche Sensoren bzw. Eingangssignale zu betätigen sind.

Abbildung 21:
Transponierte und reduzierte Darstellung der C&E-Matrix (Dokument B4)

| Aktoren | | Abschaltungen | | C1 | | | D1 | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | beteiligte SFs | Software entspricht der Matrixdokumentation | | | Funktion geprüft | | |
| Ausgang | Bezeichnung | Betriebsart Auto | | ok/n.ok | Name | Datum | ok/n. ok | Name | Datum |
| QS_M1 | Motor M1 | EMST_OK&SG1_OK & (SG2_OK v SG3_OK)<br>EMST, IS_SG1,2,3_1,2 | 1, 2, 4 | | | | | | |
| QS_M2 | Motor M2 | EMST_OK&SG2_OK<br>EMST, IS_SG2_1,2 | 1, 3 | | | | | | |
| QS_M3 | Motor M3 | EMST_OK&IS_SL_SG2<br>EMST, IS_SL_SG2 | 1, 5 | | | | | | |
| V1 | Verifikation durchgeführt (ok/n. ok): | | | Softwaresignatur: | | | | | |
| | Datum: | | | | | | | | |
| | Name: | | | | | | | | |

Abbildung 22:
Weiter komprimierte Matrix (Dokument B4)

| Aktoren | | Abschaltungen | | C1 | | | D1 | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | beteiligte SFs | Software entspricht der Matrixdokumentation | | | Funktion geprüft | | |
| Ausgang | Bezeichnung | Betriebsart Auto | | ok/n. ok | Name | Datum | ok/n. ok | Name | Datum |
| QS_M1 | Motor M1 | EMST_OK&SG1_OK & (SG2_OK v SG3_OK) | 1, 2, 4 | | | | | | |
| QS_M2 | Motor M2 | EMST_OK&SG2_OK | 1, 3 | | | | | | |
| QS_M3 | Motor M3 | EMST_OK&IS_SL_SG2 | 1, 5 | | | | | | |
| V1 | Verifikation durchgeführt (ok/n. ok): | | | Softwaresignatur: | | | | | |
| | Datum: | | | | | | | | |
| | Name: | | | | | | | | |

Die Ansteuerlogik wird komplett mit den Ausgangssignalen der Ansteuerungsebene beschrieben. Werden in der Vorverarbeitung komplexere Signale gebildet oder mehrere Signale zusammengefasst, kann es schwierig sein, alle beteiligten Eingänge aufzulisten. Dann empfiehlt sich die ganz kompakte Darstellung wie in Abbildung 22.

## 6.10 Hinweise zur Vorverarbeitungsebene

Es sei hier darauf hingewiesen, dass die Vorverarbeitungsebene nicht immer so einfach dargestellt werden kann wie in Abbildung 20. In komplexeren Fällen besteht die Projektierungsaufgabe darin, in der Vorverarbeitungsebene Signale logisch so zu verknüpfen, dass die Verknüpfungsausgangsgrößen direkt bei der Aktoransteuerung im Modul ACT verwendet werden können. Bei Vorverarbeitungsbausteinen für Schutzeinrichtungen ist auf die festgelegte negative Logik zu achten:

- Ausgangsgröße = TRUE/1 bedeutet: Schutzeinrichtung nicht ausgelöst, Gefährdungen sind vorhanden, aber abgesichert
- Ausgangsgröße = FALSE/0 bedeutet: Schutzeinrichtung hat ausgelöst, Gefährdungen sind abzuschalten

Ein Beispiel für solch eine Signalzusammenfassung zeigt Abbildung 23. Hier wird die Stillstandsüberwachung (Safe Operating Stop) mehrerer Achsen zusammengefasst zu einem Stillstandssignal. Im Beispiel der Werkzeugmaschine (Abschnitt 7.5) ist eine praktische Anwendung für diese Zusammenfassung zu sehen.

Abbildung 24 stellt ein Beispiel für die Freigabe einer zeitgesteuerten Türentriegelung in der Vorverarbeitungsebene dar. Die Aufforderung zur Türentriegelung (Taster I_OP_SG) setzt ein Flip-Flop. Nach dem Ablauf der parametrierbaren Zeit TIME wird das Entriegelungssignal S_UNLOCK ausgegeben. Die vollständige Öffnung der Tür setzt den Timer F_TON zurück. Hier wäre es sinnvoll, die Logik in Abbildung 24 in einem wiederverwendbaren Bibliotheksbaustein zu kapseln.

Die Übergangszeit bis zur Öffnung der Tür kann in der Matrix nicht abgelesen werden. Beim Funktionstest muss diese Information aus der Definition der Sicherheitsfunktion abgelesen werden.

Durch die Verlagerung von zeitlichen Aspekten und Verknüpfung komplexer Signale in die Vorverarbeitungs- oder Ansteuerungsebene ist es meistens möglich, die Ansteuerlogik ACT auf rein binäre Verknüpfungen (UND, ODER, NICHT) zu begrenzen.

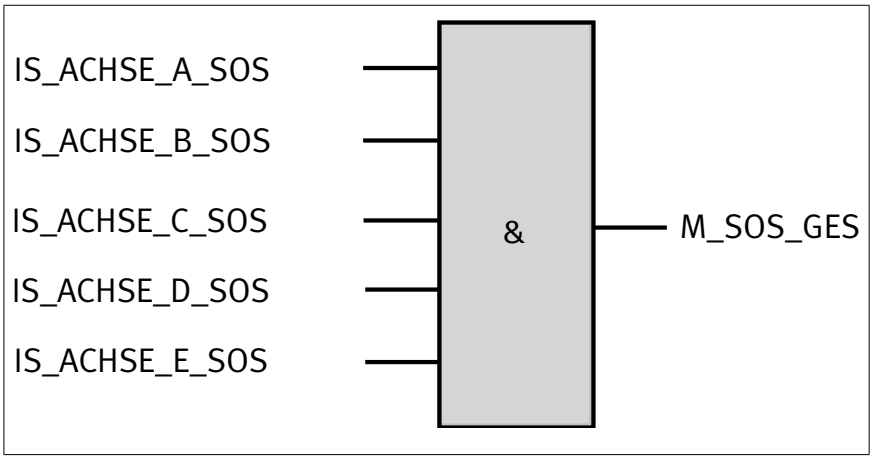



Abbildung 23:
Zusammenfassung von Stillstandssignalen in der Vorverarbeitungsebene

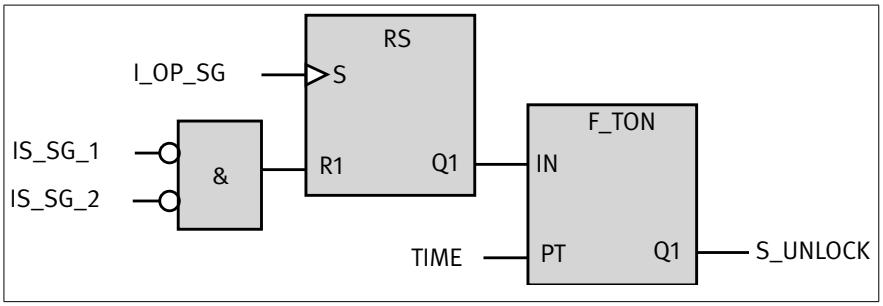


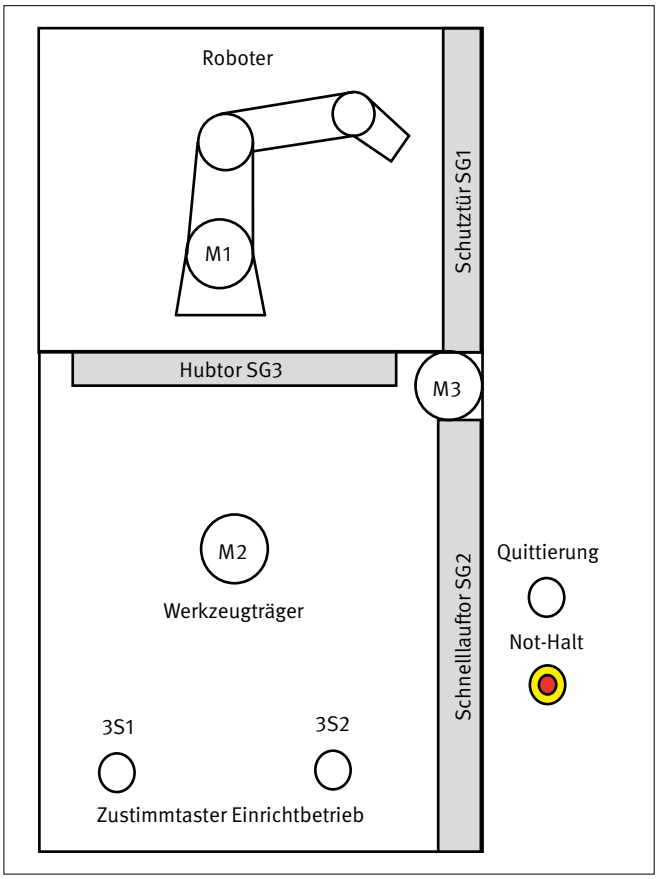

Abbildung 24:
Beispiel einer zeitgesteuerten Türentriegelung in der Vorverarbeitungsebene

## 6.11 Berücksichtigung mehrerer Betriebsarten und eigener Funktionsbausteine

Das obige Beispiel wird nun um einen Einrichtbetrieb erweitert (entsprechend dem Beispiel in Abschnitt 7.2). An diesem Beispiel ist die Unterscheidung der Betriebsarten deutlich zu erkennen. Außerdem wird in diesem Beispiel die Ansteuerung eines Motors mit sicher begrenzter Geschwindigkeit (SLS = Safely-Limited Speed) mit einem selbstentwickelten Funktionsbaustein realisiert. Die Entwicklung eines solchen Funktionsbausteins ist in Abschnitt 6.12 beschrieben.

Abbildung 25 zeigt die Anlagenskizze (Dokument A2.1) der Roboterzelle mit integriertem Einrichtbetrieb für die Achse M2.

Abbildung 25:
Anlagenskizze (Dokument A2.1) Roboterzelle mit Einrichtbetrieb




Zusätzlich zur Funktion aus dem vorherigen Beispiel kommt hinzu, dass der Werkzeugträger (M2) bei geöffnetem Schnelllauftor und geschlossenem Hubtor im Einrichtbetrieb über zwei an verschiedenen Stellen angeordnete Zustimmtaster mit sicher begrenzter Drehzahl verfahren werden kann. Sobald die Geschwindigkeit des Motors zu groß wird, schaltet er sicher ab.

Damit ergeben sich aus der Risikobeurteilung sieben Sicherheitsfunktionen. Die ersten fünf bleiben unverändert aus dem vorherigen Beispiel erhalten. Tabelle 13 zeigt die Sicherheitsfunktionen der Roboterzelle mit Einrichtbetrieb.

Abbildung 26 zeigt den Stromlaufplan für dieses Beispiel. Geändert hat sich nur, dass die beiden Tipptaster 3S1 und 3S2 hinzugekommen sind. Darüber hinaus wird der Motor M2 von einem Umrichter mit integrierten Sicherheitsfunktionen angesteuert.

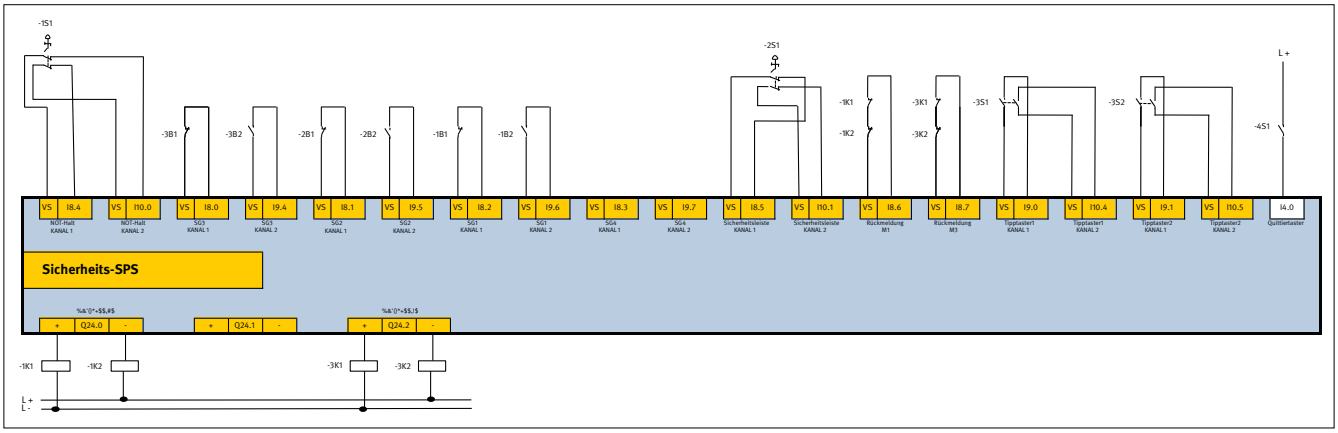
Da in diesem Beispiel ein Antrieb mit integrierten Sicherheitsfunktionen verwendet wird, hat sich der Systemaufbau (Dokument A2.3) deutlich verändert. Hier ist nun ein Antrieb mit Feldbuskopplung hinzugekommen. Abbildung 27 zeigt den Systemaufbau.

Tabelle 14 (Seite 50) zeigt die Symboltabelle dieses Beispiels. Änderungen gegenüber dem vorherigen Beispiel ergeben sich durch die beiden Tipptaster bei den Ausgängen und die Ansteuerung der Sicherheitsfunktionen des Umrichters bei den Ausgängen.

Tabelle 13:
Sicherheitsfunktionen (Dokument A1) der Roboterzelle mit Einrichtbetrieb

| Nr. | Beschreibung | $PL_r$ | Reaktionszeit in ms | Priorität | Betriebsart |
|---|---|---|---|---|---|
| SF1 | Wenn der Not-Halt EMST betätigt wird, werden M1, M2 und M3 abgeschaltet. | d | 100 | 1 | Alle |
| SF2 | Wenn Schutztür SG1 auf, dann wird M1 abgeschaltet. | d | 100 | 1 | Alle |
| SF3 | Wenn Schnelllauftor SG2 auf, dann wird M2 abgeschaltet. | d | 100 | 2 | Automatikbetrieb |
| SF4 | Wenn Schnelllauftor SG2 auf und Hubtor SG3 auf, dann wird M1 abgeschaltet. | d | 100 | 1 | Alle |
| SF5 | Wenn die Sicherheitsleiste SL_SG2 des Schnelllauftors SG2 betätigt wird, dann wird der Motor M3 abgeschaltet. | d | 100 | 1 | Alle |
| SF6 | Wenn Schnelllauftor SG2 offen und Hubtor SG3 geschlossen und der Tipptaster 3S1 betätigt ist, so ist SLS für M2 freigegeben (SLS aktiv). | d | 100 | 2 | Einrichtbetrieb |
| SF7 | Wenn Schnelllauftor SG2 offen und Hubtor SG3 geschlossen und der Tipptaster 3S2 betätigt ist, so ist SLS für M2 freigegeben (SLS aktiv). | d | 100 | 2 | Einrichtbetrieb |

Abbildung 26:
Stromlaufplan (Dokument A2.2) der Roboterzelle mit Einrichtbetrieb



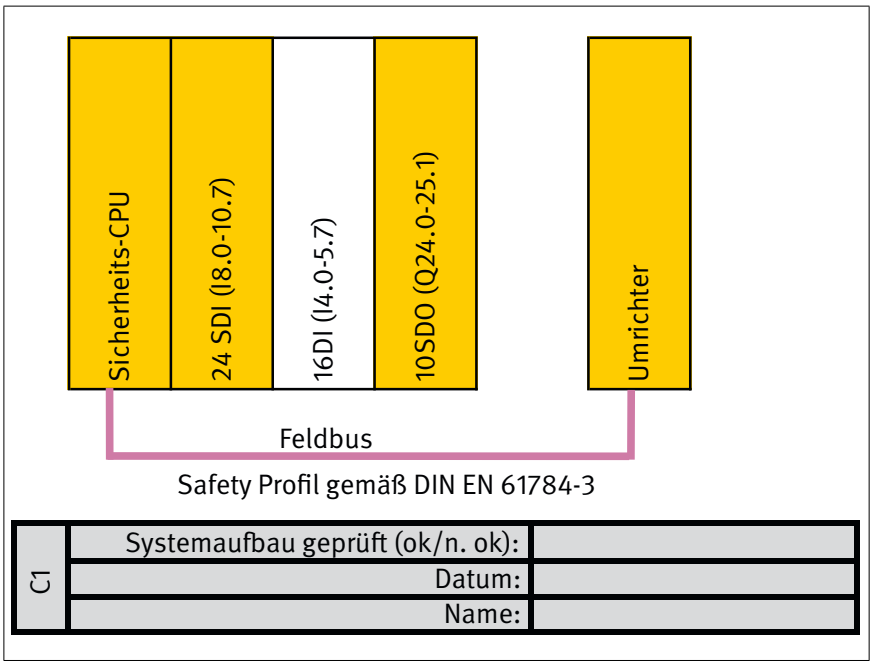




Abbildung 27:
Systemaufbau (Dokument A2.3) der Roboterzelle mit Einrichtbetrieb

Tabelle 14:
Symboltabelle (Dokument A2.4) der Roboterzelle mit Einrichtbetrieb

| Signale | Variable | Adresse | D1 | C1 |
|---|---|---|---|---|
| | | | Validiert (ok/n. ok) | richtige Verschaltung in der Software verifiziert (ok/n. ok) |
| **Eingänge** | | | | |
| Not-Halt, zweikanalig (NC) (1S1, 1S2, 1S3) | IS_EMST | %I8.4 | | |
| Kontakt 1 Schutztür SG1 (NC) (1B1) | IS_SG1_1 | %I8.2 | | |
| Kontakt 2 Schutztür SG1 (NO) (1B2) | IS_SG1_2 | %I9.6 | | |
| Kontakt 1 Schnelllauftor SG2 (NC) (2B1) | IS_SG2_1 | %I8.1 | | |
| Kontakt 2 Schnelllauftor SG2 (NO) (2B2) | IS_SG2_2 | %I9.5 | | |
| Kontakt 1 Hubtor SG3 (NC) (3B1) | IS_SG3_1 | %I8.0 | | |
| Kontakt 2 Hubtor SG3 (NO) (3B2) | IS_SG3_2 | %I9.4 | | |
| Sicherheitsleiste von SG2, zweikanalig (NC) (2S1) | IS_SL_SG2 | %I8.5 | | |
| Rückmeldung Schütze M1 (NC) (1K1, 1K2) | IS_SM1 | %I8.6 | | |
| Rückmeldung Schütze M3 (NC) (3K1, 3K2) | IS_SM3 | %I8.7 | | |
| Tipptaster 1 SLS, zweikanalig (NO) (3S1) | IS_TIP_1 | %I9.0 | | |
| Tipptaster 2 SLS, zweikanalig (NO) (3S2) | IS_TIP_2 | %I9.1 | | |
| Sicherheitsfehlermeldung FU | IS_Err_FU | %I32.7 | | |
| Quittiertaster (NO) (4S1) | I_ACK | %I4.0 | | |
| **Ausgänge** | | | | |
| Schütze Motor M1 (1K1, 1K2) | QS_M1 | %Q24.0 | | |
| Schütze Motor M3 (3K1, 3K2) | QS_M3 | %Q24.2 | | |
| Aktivierung von STO für M2 | QS_M2_STO | %Q32.0 | | |
| Aktivierung von SLS für M2 | QS_M2_SLS | %Q32.4 | | |
| Aktivierung von SS1 für M2 | QS_M2_SS1 | %Q32.1 | | |
| Aktivierung von SS2 für M2 | QS_M2_SS2 | %Q32.2 | | |
| Aktivierung von SOS für M2 | QS_M2_SOS | %Q32.3 | | |
| Quittierung FU Sicherheitsfehler | QS_M2_ACK_FU | %Q32.7 | | |
| | | Datum: | | |
| | | Name: | | |
| | | Softwaresignatur: | | |

An dem Katalog fehlervermeidender Maßnahmen (Dokument A3), den normativen Anforderungen (A4), der Architektur des Sicherheitsprogramms (B1), der Architektur des Standardprogramms (B2) und des Codereviews (C1) ergeben sich keine Änderungen zum vorherigen Beispiel.

Abbildung 28 zeigt die Modularchitektur (Dokument B3) des Beispiels. Hier ist erstmals in diesen Beispielen ein vom Programmierenden selbst zu entwickelnder Funktionsbaustein (EN_SLS) enthalten.

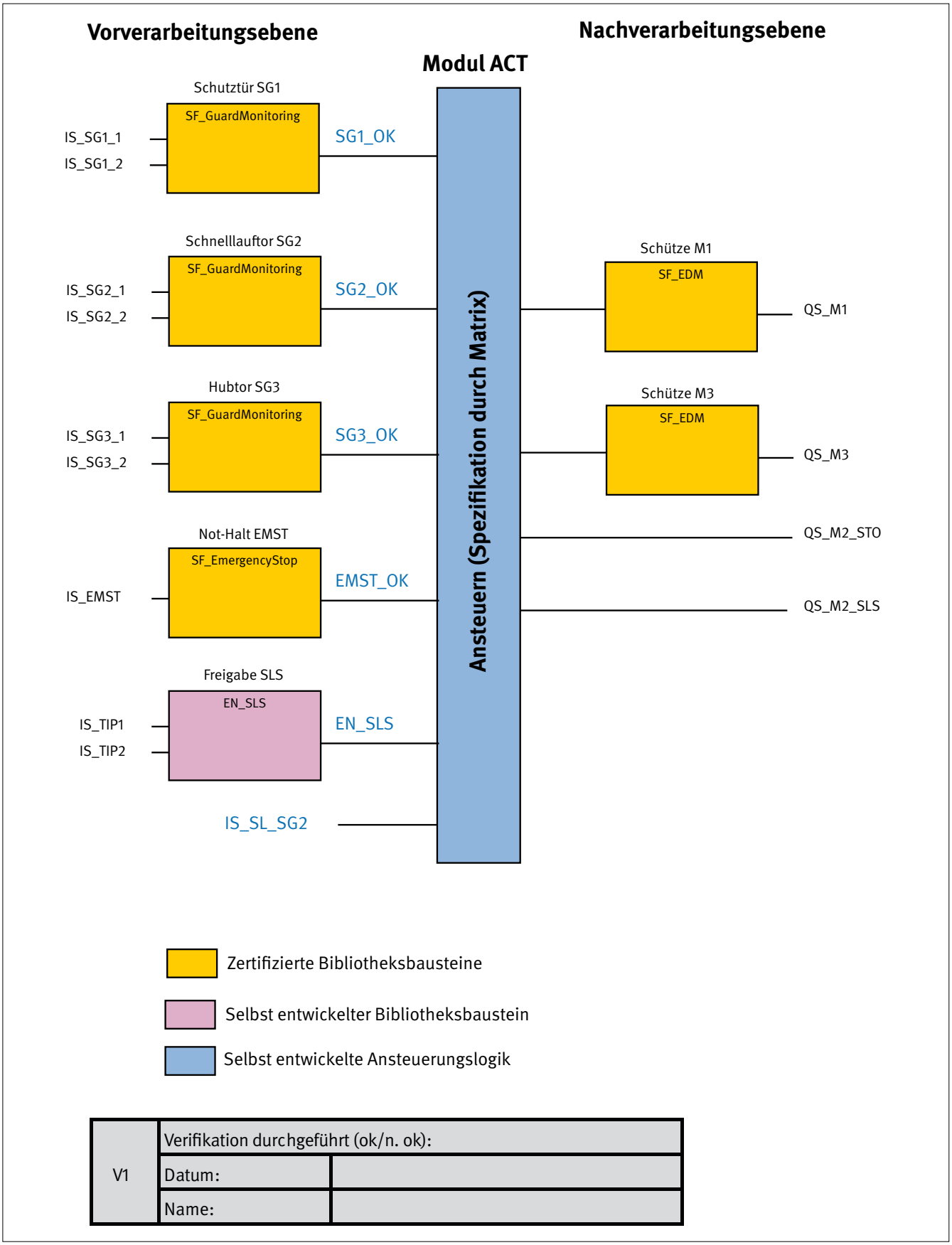
In der Modularchitektur sind die Ausgänge für den sicheren Umrichter des Motors M2 zu sehen. Im Folgenden werden diese Befehle kurz erläutert (siehe DIN EN 61800-5-2):

- STO (Safe Torque Off): Bei Aktivierung des Befehls STO wird die Energiezufuhr zum Antrieb sicher abgeschaltet und der Antrieb trudelt aus.
- SLS (Safely-Limited Speed): Der Befehl SLS dient der Überwachung einer festgesetzten Maximalgeschwindigkeit, wobei die Überwachung unabhängig von der Drehrichtung ist. Wird die Drehzahl überschritten, wird STO automatisch im Umrichter aktiviert.
- SS1 (Safe Stop 1): Der Befehl SS1 bewirkt einen Stopp der Kategorie 1 nach DIN EN 60204-1. Der Antrieb wird in einer parametrierbaren Zeit abgebremst und danach wird mit STO die Energiezufuhr abgestellt.
- SS2 (Safe Stop 2): Beim Befehl SS2 wird der Antrieb ebenfalls mit einer parametrierbaren Zeit abgebremst. Danach wird jedoch mit SOS der sichere Stillstand des Antriebs überwacht.
- SOS (Safe Operating Stop): Der Befehl SOS dient zur sicheren Überwachung des Stillstands des Antriebs. Dreht der Motor trotzdem, wird mit STO die Energiezufuhr zum Antrieb abgeschaltet.
- Alle Sicherheitsfunktionen werden mit einem FALSE/0-Signal aktiviert (negative Logik). Bei Ausfall der Kommunikation mit der Steuerung werden so alle Sicherheitsfunktionen aktiv

und der Antrieb stoppt. Daher werden alle nicht verwendeten Sicherheitsfunktionen (SS1, SS2, SOS) mit dem Signal TRUE deaktiviert.

- Der Ausgang QS_M2_ACK_FU dient zur Quittierung von Fehlern im Umrichter.

Abbildung 28:
Modularchitektur (Dokument B3) der Roboterzelle mit Einrichtbetrieb




| | | |
|---|---|---|
| V1 | Verifikation durchgeführt (ok/n. ok): | |
| | Datum: | |
| | Name: | |

Den Funktionsbaustein EN_SLS gilt es zu spezifizieren. Mit dem Signal „EN_SLS“ wird der Ausgang QS_M2_SLS mit dem Signal FALSE aktiviert. Da die Sicherheitsfunktionen des Umrichters alle mit FALSE aktiviert werden, ist es sinnvoll, den Ausgang des Funktionsbausteins auch in negativer Logik zu definieren.

In Abbildung 29 ist die C&E-Matrix (Dokument B4) der Roboterzelle mit Einrichtbetrieb zu sehen. Zur besseren Übersicht sind die Zustände der Eingangssignale nicht dargestellt. An den ersten fünf Sicherheitsfunktionen hat sich nichts geändert. Für den Einrichtbetrieb ist ein Grundzustand definiert (Zustand 7, weiß hinterlegt). Von diesem Zustand aus kann der Motor M2 mit sicher begrenzter Geschwindigkeit verfahren werden. In dieser Matrix ist zusätzlich eine Prüfzeile mit einem Testfall enthalten (Zustand 7).

Für weitere Tests können beliebig viele Zeilen mit Testfällen in der C&E-Matrix ergänzt werden (hier gelb unterlegt).

Abbildung 29:
C&E-Matrix (Dokument B4) der Roboterzelle mit Einrichtbetrieb

| | | | Cause | Effect | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Ausgänge | | | | | D1 | | |
| Betriebsart | Vorgängerzustand bei Test | Zustand | Bezeichnung | QS_M1 (Q24.0) | QS_M2_STO (Q32.0) | QS_M2_SLS (Q32.4) | QS_M3 (Q24.2) | Quittierung (I_ACK (I4.0) | Validiert (ok/n. ok) | Name | Datum |
| | | | **C1: Software entspricht Matrixnotation** | | | | | | | | |
| | | 1 | **ALL_OK** | EIN | EIN | EIN | EIN | | | | |
| | 1 | 2 | SF1: Not-Halt betätigt | EMST_OK<br>AUS | EMST_OK<br>AUS | NOP | EMST_OK<br>AUS | EIN | | | |
| | 1 | 3 | SF2: SG1 offen | SG1_OK<br>AUS | NOP | NOP | NOP | EIN | | | |
| | 1 | 4 | SF3: SG2 offen | NOP | SG2_OK&E<br>N_SLS<br>AUS | NOP | NOP | EIN | | | |
| | 1 | 5 | SF4: SG2 und SG3 offen | SG2_OK v<br>SG3_OK<br>AUS | NOP | NOP | NOP | EIN | | | |
| | 1 | 6 | SF5: Sicherheitsleiste betätigt | NOP | NOP | NOP | IS_SL_SG2<br>AUS | EIN | | | |
| | | 7 | SG2 offen, SG3 geschlossen,<br>IS_TIP1, 2 nicht betätigt | NOP | AUS | EIN | NOP | | | | |
| | 7 | 8 | SF6: SG2 offen, SG3 geschlossen,<br>IS_TIP_1 betätigt | NOP | /EN_SLS<br>EIN | EN_SLS<br>AUS | NOP | EIN | | | |
| | 7 | 9 | SF7: SG2 offen, SG3 geschlossen,<br>IS_TIP_2 betätigt | NOP | /EN_SLS<br>EIN | EN_SLS<br>AUS | NOP | EIN | | | |
| | 7 | 10 | SG2 offen, SG3 geschlossen,<br>IS_TIP_1, 2 betätigt | NOP | AUS | EIN | NOP | | | | |
| V1 | Verfikation durchgeführt (ok/n. ok): | | | | | Softwaresignatur: | | | | | |
| | Datum: | | | | | | | | | | |
| | Name: | | | | | | | | | | |

Automatikbetrieb
Einrichtbetrieb
Alle

In dieser Matrix tauchen zum ersten Mal Negationen der Variablen auf, die durch einen vorangestellten Schrägstrich gekennzeichnet werden.

Erinnert sei nochmals an das Bildungsgesetz für das Ansteuermodul ACT:

Schritt 1: Für jede einzelne Sicherheitsfunktion, die bei einem Aktor einen Schaltvorgang auslöst (bezogen auf den Vorgängerzustand), trägt man betriebsartenabhängig in die entsprechende Zelle der Tabelle diejenige logische Verknüpfung der Eingangsgrößen von ACT ein, die den Schaltvorgang auslöst. Dieser Schaltvorgang wird hier durch „AUS“ bzw. „EIN“ angegeben. Löst eine Sicherheitsfunktion bei einem Aktor keinen Schaltvorgang aus, ist dort ein „NOP“ einzutragen.

Es folgen mehrere Beispiele für die Sicherheitsfunktionen SF3, SF4 und SF5 aus Abbildung 29:

- Wenn SF3 in der Betriebsart „Automatik“ über das Öffnen von SG2 aktiviert wird, wird M2 (bezogen auf den Vorgängerzustand All_OK) über STO abgeschaltet, sofern nicht für M2 die sicher begrenzte Geschwindigkeit (SLS) aktiviert wurde. Daher muss man in diese Zelle SG2_OK & EN_SLS eintragen.

- Wenn SF4 in der Betriebsart „Alle“ über das Öffnen von SG2 und SG3 aktiviert wird, wird M1 abgeschaltet. Daher muss man in diese Zelle SG2_OK v SG3_OK eintragen. Bei QS_M2_STO muss NOP eingetragen werden, da M2 von dieser Sicherheitsfunktion nicht in dieser Betriebsart tangiert wird.

- Wenn SF6 in der Betriebsart „Einrichtbetrieb“ bei offener Schutztür SG2 und geschlossener Schutztür SG3 durch das Drücken des Tipptasters IS_TIP_1 aktiviert wird, wird QS_M2_STO eingeschaltet (EIN: d. h. hier STO deaktiviert) und QS_M2_SLS ausgeschaltet (AUS: d. h. hier SLS freigegeben). Daher muss bei QS_M2_STO die negierte Variable /EN_SLS

und bei QS_M2_SLS die Variable EN_SLS (Freigabe durch EN_SLS = FALSE) eingetragen werden.

Schritt 2: Die Eingänge der UND-Glieder in Abbildung 16 ergeben sich folgendermaßen aus Abbildung 29: Pro Ausgang und pro Betriebsart müssen nur die eingetragenen Variablen mit UND verknüpft werden.

In Abbildung 30 ist die Programmskizze zu sehen. Deutlich ist die Struktur des Ansteuermoduls aus Abbildung 16 im Modul ACT am Ausgang QS_M2_STO zu erkennen. Am rechten UND steht die Bedingung für alle Betriebsarten (EMST_OK), links davor ein ODER mit der Betriebsart Automatikbetrieb (SG2_OK & EN_SLS) und der Betriebsart Einrichtbetrieb (NOT (EN_SLS)).

Die Sicherheitsfunktion SLS kann auch in Kombination mit der Sicherheitsfunktion SS2 genutzt werden. Dann bleibt der Antrieb in Regelung und wird auf sicheren Stillstand mit SOS überwacht, wenn kein Tipptaster betätigt ist. Dies ist der Einfachheit halber hier nicht umgesetzt.

Der Vollständigkeit halber zeigt Abbildung 31 die kompakte Darstellung der Matrix.

Damit weitere sinnvolle Tests auch mit der kompakten Form möglich bleiben, wird hier eine zusätzliche Tabelle mit Testfällen angelegt. Abbildung 32 zeigt die Einträge für zusätzliche Testfälle.

Abbildung 30:
Programmskizze der Roboterzelle mit Einrichtbetrieb

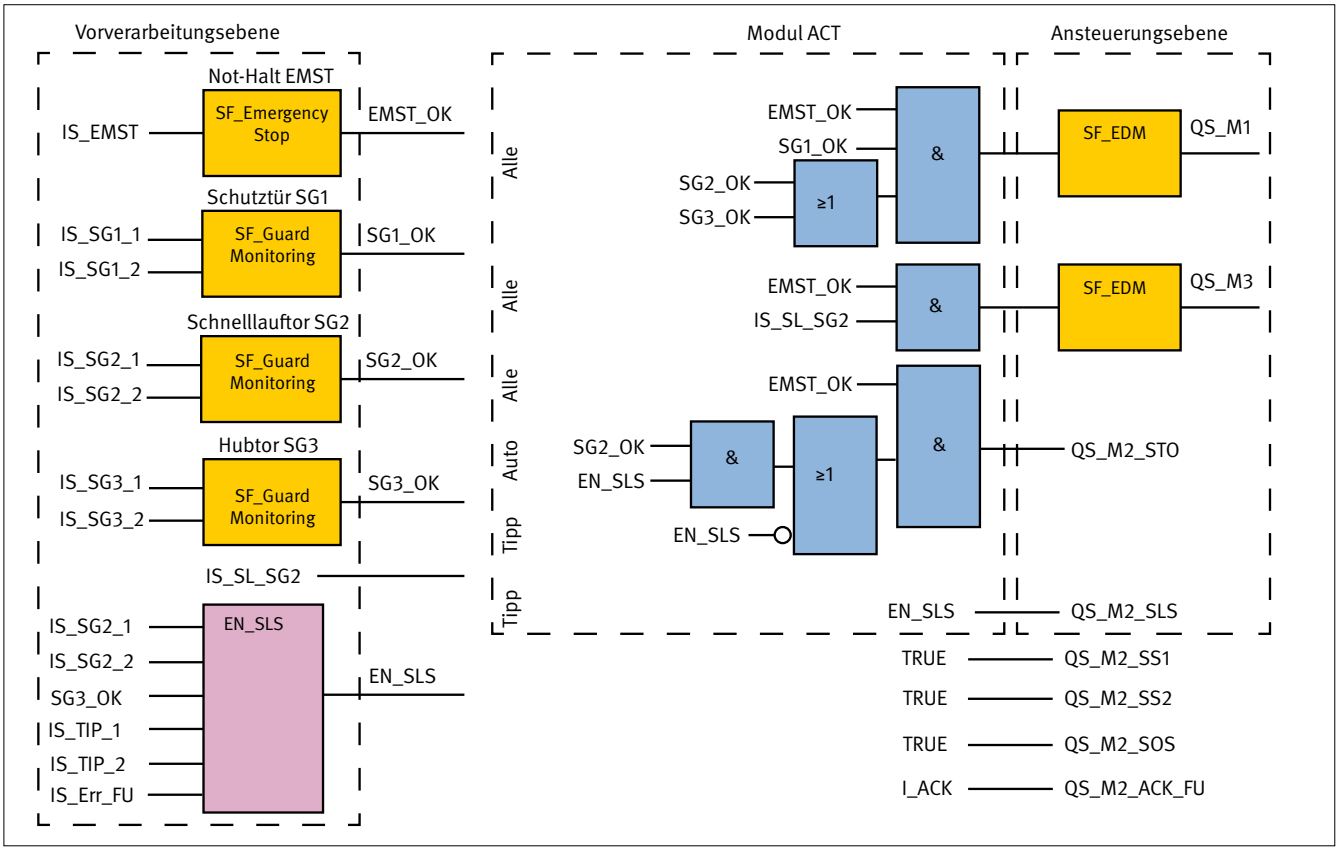



Abbildung 31:
Kompakte Matrixdarstellung für das Beispiel der Roboterzelle mit Einrichtbetrieb

| Aktoren | | Abschaltungen | | | | C1 | | | D1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | beteiligte SFs | Software entspricht Matrixdokumentation | | | Funktion validiert | | |
| Ausgang | Bezeichnung | Betriebsart alle | Automatik-betrieb | Einricht-betrieb | | ok/n. ok | Name | Datum | ok/n. ok | Name | Datum |
| QS_M1 | motor M1 | EMST_OK & SG1_OK & (SG2_OK v SG3_OK) | | | 1, 2, 4 | | | | | | |
| QS_M2_STO | motor M2 STO | EMST_OK | SG2_OK & EN_SLS | /EN_SLS | 1, 3, 6, 7 | | | | | | |
| QS_M2_SLS | motor M2 SLS | | | EN_SLS | 6, 7 | | | | | | |
| QS_M3 | motor M3 | EMST_OK & IS_SL_SG2 | | | 1, 5 | | | | | | |
| V1 | Verifikation durchgeführt (ok/n. ok): | | | | | Softwaresignatur: | | | | | |
| | Datum: | | | | | | | | | | |
| | Name: | | | | | | | | | | |

Abbildung 32:
Zusätzliche Testfälle für das Beispiel der Roboterzelle mit Einrichtbetrieb

| Zusätzliche Testfälle | | D1 | | |
|---|---|---|---|---|
| | | Validiert | | |
| Testfall | Reaktion | ok/n. ok | Name | Datum |
| SG2 offen, SG3 geschlossen, beide Tipptaster 3S1 und 3S2 betätigt | SLS nicht freigegeben | | | |

## 6.12 Behandlung konfigurierbarer Sicherheitssteuerungen

Neben den mit einer typischen SPS-Sprache (FBD, LD, z. B. nach DIN EN ISO 61131-3 [12]) frei programmierbaren SPSen können Sicherheitsfunktionen zunehmend auch mit kompakten, „grafisch konfigurierbaren“ Steuerungen realisiert werden. Es stellt sich die Frage, inwieweit hier noch die normativen Anforderungen für SRASW anwendbar sind, da z. B. DIN EN ISO 13849-1 in Abschnitt 4.6.4 eigene Anforderungen für „Softwarebasierende Parametrisierung“ beschreibt. Das Normengremium hat in diesem Abschnitt aber an Steuerungsgeräte wie Sensoren oder Antriebssteuerungen gedacht, deren Funktionen zwar parametriert, aber nicht logisch verknüpft werden können.

Typisch für die Gattung konfigurierbarer Steuerungen sind Programmiertools mit einfacher grafischer Oberfläche. In einem grafischen Editor können Funktionsbausteine platziert und untereinander bzw. mit Ein- und Ausgängen verbunden werden. Abschließend sind die Funktionsbausteine zu parametrieren – eigentlich genauso wie bei einer klassischen SPS, nur können keine textuellen Sprachen benutzt werden. Auch wenn diese Steuerungen als einfach und schnell „konfigurierbar“ beworben werden: Auch hier können bei der logischen Verschaltung Fehler entstehen wie bei den „großen“ Steuerungen.

Daher sind die Programme dieser Steuerungen als SRASW anzusehen und können mit der Matrixmethode des IFA spezifiziert, validiert und dokumentiert werden. Ein Beispiel dafür findet sich in Abschnitt 7.10.

## 6.13 Matrixbasierte Dokumentation von eigenen Funktionsbausteinen

Aus dem Beispiel der Roboterzelle mit Einrichtbetrieb soll nun anhand des Funktionsbausteins „EN_SLS“ die Vorgehensweise zur Spezifikation von eigenen Funktionsbausteinen mit der Matrixmethode des IFA beschrieben werden.

Zur Spezifikation des Funktionsbausteins „EN_SLS“ wird das vereinfachte V-Modell zur Entwicklung von Funktionsbausteinen (Abbildung 7, siehe Abschnitt 5.4) genutzt. Die zugehörigen Dokumente sind in Tabelle 2, Abschnitt 5.5, aufgelistet.

Abbildung 33 zeigt die Schnittstellenbeschreibung (Dokument AM1) für den Funktionsbaustein. Sie enthält eine Funktionsbeschreibung und alle Ein- und Ausgänge werden mit symbolischer Bezeichnung, Datentyp und Kommentar angegeben.

Der Katalog fehlervermeidender Maßnahmen für einen eigenen Funktionsbaustein ist etwas kompakter als der für das gesamte Anwendungsprogramm. Die Tabellen 15 und 16 zeigen beispielhaft die allgemeinen und steuerungsspezifischen Maßnahmen zur Entwicklung von Funktionsbausteinen (Dokument AM2).

Abbildung 33:
Schnittstellenbeschreibung (Dokument AM1) des Funktionsbausteins EN_SLS

| **FB Name: EN_SLS** | | | |
|---|---|---|---|
| **Funktionsbeschreibung** | | | |
| Der Funktionsblock EN_SLS generiert das Freigabesignal EN_SLS für die sicher begrenzte Geschwindigkeit (SLS). Die Freigabe erfolgt mit einer logischen 0. Damit die Freigabe erteilt wird, muss die Schutztür SG1 offen sein (0-Signal an bSG1_1 und bSG1_2), die Schutztür SG2 geschlossen und quittiert sein (1-Signal an bSG2), es darf kein Fehler im Frequenzumrichter anliegen (0-Signal an bERROR), nur ein Freigabetaster (bTIP_1 oder bTIP_2) ist betätigt. Um einen automatischen Wiederanlauf zu verhindern, werden die positiven Flanken der beiden Taster ausgewertet. | | | |
| **Eingänge** | | | |
| **Name** | **Datentyp** | **Anfangswert** | **Beschreibung, Parameterwerte** |
| bSG1_1 | SAFEBOOL | FALSE | Kontakt 1 von SG1. FALSE: Tür offen. TRUE: Tür geschlossen. |
| bSG1_2 | SAFEBOOL | FALSE | Kontakt 2 von SG1. FALSE: Tür offen. TRUE: Tür geschlossen. |
| bSG2 | SAFEBOOL | FALSE | Ausgang von SF_Guardmonitoring FB für SG2. FALSE: Tür offen. TRUE: Tür geschlossen. |
| bTIP_1 | SAFEBOOL | FALSE | Freigabetaster 1. FALSE: Taster nicht betätigt. TRUE: Taster betätigt. |
| bTIP_2 | SAFEBOOL | FALSE | Freigabetaster 2. FALSE: Taster nicht betätigt. TRUE: Taster betätigt. |
| bERROR | SAFEBOOL | FALSE | Umrichterfehler. FALSE: Kein Fehler. TRUE: Fehler. |
| **Ausgänge** | | | |
| bEN_SLS | SAFEBOOL | TRUE | Freigabesignal für SLS. FALSE: Freigabe erteilt. TRUE: Keine Freigabe für SLS. |
| Hinweise: Keine | | | |

Tabelle 15:
Beispielkatalog allgemeiner fehlervermeidender Maßnahmen zur Entwicklung eigener Funktionsbausteine

| | | CM1 | |
|---|---|---|---|
| | | **Kürzel** | **realisiert (J/N)** |
| **A. Variablen** | | | |
| Präfixe für boolesche Variable: „b“. | | RMA1 | |
| Präfixe für Instanzen: Timer: „T_“; positive Flankenerkennung: „R_“; Flip-Flops: “FF_” | | RMA2 | |
| Variablennamen: Der Variablenname nach dem Präfix sollte selbsterklärend sein, z. B. sollte der Name der relevanten Funktionseinheit enthalten sein. Beispiel: ..SG1.. für die Sicherheitstür SG1. | | RMA3 | |
| Variablendeklaration: Initialisierung mit dem sichersten Wert. Jede Deklaration enthält einen Kommentar. | | RMA4 | |
| Interface: Jeder Funktionsbaustein kommuniziert mit der Umgebung ausschließlich über die Input/Output-variablen. | | RMA5 | |
| Globale Variable: Sind nicht erlaubt. | | RMA6 | |
| **B. Signalverarbeitung** | | | |
| Zuweisungen: Variablen werden in nur einem Programmstatement zugewiesen. | | RMB1 | |
| Kommentare: Jedes Netzwerk enthält einen Kommentar. | | RMB2 | |
| Schutz: Der Funktionsbaustein ist durch ein Passwort geschützt. | | RMB3 | |
| **C. Bibliotheksbausteine** | | | |
| Verwendung: Wo immer möglich, sollen Bibliotheksbausteine verwendet werden. | | RMC1 | |
| | Datum: | | |
| | Name: | | |
| | Softwaresignatur: | | |

Tabelle 16:
Beispielkatalog steuerungsspezifischer fehlervermeidender Maßnahmen zur Entwicklung eigener Funktionsbausteine

| | | | CM1 | |
|---|---|---|---|---|
| | | | Kürzel | realisiert (J/N) |
| Programmeditor/Programmiersprache | | | | |
| Genutzter Programmeditor | Safety Editor V10.1 | | RS1 | |
| Programmiersprache | Funktionsbausteinsprache (FBD) | | RS2 | |
| Softwarebibliothek | Safety Library V3.2 | | RS3 | |
| | | Datum: | | |
| | | Name: | | |
| | | Softwaresignatur: | | |

Abbildung 34 zeigt die Modulspezifikation mit den grauen Verifikations- und Validierungsfeldern (Dokument BM1) für den Funktionsbaustein EN_SLS. Hier wird nochmals kurz das Verhalten des Ausgangs beschrieben und in Form einer C&E-Matrix dargestellt. Die C&E-Matrix dient wieder als Testgrundlage und zur Spezifikation der Software. Das Modell des Funktionsbausteins EN_SLS besteht aus einem „setzen dominanten“ Flip-Flop, das durch die in Abbildung 34 angedeuteten ODER-Kombinationen gesetzt bzw. zurückgesetzt wird.

Abbildung 34:
Modulspezifikation und Testplan des Funktionsbausteins EN_SLS (Dokument BM1), EQ = EQUIVALENT, / = NOT, › = positive Flanke

**Beschreibung**

EN_SLS wird durch ein “setzen dominantes” Flip-Flop realisiert. Das Flip-Flop wird durch eine ODER-Verknüpfung gesetzt. Das Rücksetzen erfolgt wieder durch eine ODER-Verknüpfung. Die Eingänge dieser ODER-Verknüpfungen werden unten durch die in der Spalte „effect“ aufgeführten farbigen Variablen spezifiziert. Der erwartete Wert des Ausgangs des Funktionsbausteins ist ebenfalls in der Spalte „effect“ dargestellt.

| | | | cause | | | | | | | effect | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Eingänge | | | | | | Beschreibung | Ausgang | DM | | |
| Flip-Flop | Vorgänger zustand | Zustand | bSG1_1 | bSG1_2 | bSG2 | bTIP_1 | bTIP_2 | berror | | bEN_SLS | Korrekt (J/N) | Name | Datum |
| | | | | | | | | | **C1: Software stimmt mit Matrixspezifikation überein** | | | | |
| | | 1 | 0 | 0 | 1 | 0 | 0 | 0 | bTIP_1, 2 FALSE, SG1 offen, SG2 zu, berror FALSE | ON | | | |
| reset | | | | | | | | | bTIP_1 TRUE, | ›bTIP_1 | | | |
| | 1 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | | **OFF** | | | |
| | | | **ODER** | | | | | | bTIP_2 TRUE | ›bTIP_2 | | | |
| | 1 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | | **OFF** | | | |
| set | | | | | | | | | bSG1_1 TRUE | bSG1_1 | | | |
| | 2 | 4 | 1 | 0 | 1 | 1 | 0 | 0 | | **ON** | | | |
| | | | **ODER** | | | | | | bSG1_2 TRUE | bSG1_2 | | | |
| | 2 | 5 | 0 | 1 | 1 | 1 | 0 | 0 | | **ON** | | | |
| | | | **ODER** | | | | | | bSG2 FALSE | /bSG2 | | | |
| | 2 | 6 | 0 | 0 | 0 | 1 | 0 | 0 | | **ON** | | | |
| | | | **ODER** | | | | | | berror TRUE | berror | | | |
| | 2 | 7 | 0 | 0 | 1 | 1 | 0 | 1 | | **ON** | | | |
| | | | **ODER** | | | | | | bTIP_1 TRUE,FALSE; bTIP_2 TRUE, FALSE | bTIP_1 EQ bTIP_2 | | | |
| | 2 | 8 | 0 | 0 | 1 | 0 | 0 | 0 | | **ON** | | | |
| | | | | | | | | | **VM1** | **Softwaresignatur:** | | | |
| | | | | | | | | | | **Verifikation korrekt (J/N):** | | | |

Zur Programmierung werden die durch die farbigen Variablen dargestellten Verknüpfungen jeweils für den Setz- und Rücksetzeingang des Flip-Flops mit ODER verknüpft.

Abbildung 35 zeigt die Programmskizze (Dokument BM2) des Moduls EN_SLS. Hieraus lässt sich die Systematik gut erkennen. Tabelle 17 zeigt das Codereview (Dokument CM1) für einen selbstentwickelten Funktionsbaustein.

Das Protokoll Modultest (Dokument DM1) ergibt sich aus den Validierungsspalten rechts in Abbildung 34 (Dokument BM1).

Mit dieser Methode lassen sich einfache Funktionsbausteine (d. h. im Wesentlichen aus Flip-Flops sowie UND- und ODER-Verknüpfungen) definieren und testen. Der Test geschieht meist in der Simulation. Die Setz- und Rücksetzlogik von EN_SLS ist relativ einfach, sodass diese über eine Matrixnotation wie in Abbildung 34 (inkl. Testplan) beschrieben werden kann. Sind die zu spezifizierenden Funktionsbausteine komplizierter, so ist die Spezifikation über eine Matrix nicht mehr möglich. Für diese Fälle sei auf die von PLCopen entwickelte Spezifikationsmethodik verwiesen [8].

Abbildung 35:
Programmskizze (Dokument BM2) des Moduls EN_SLS

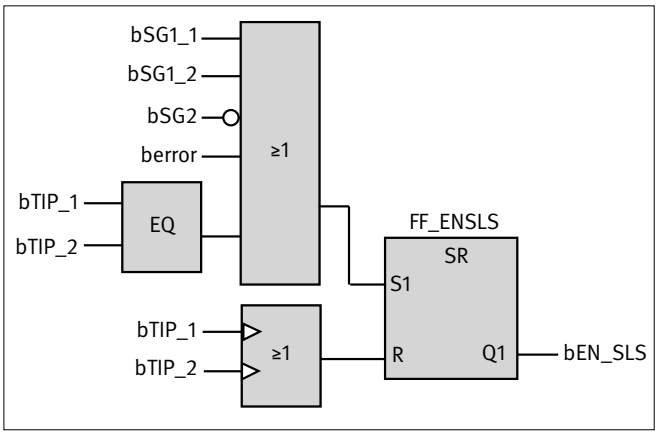



Tabelle 17:
Codereview (Dokument CM1) für eigene Funktionsbausteine

| Verifikationen | Referenz | Ja/ Nein |
|---|---|---|
| 1. Sind die vereinbarten Programmierregeln eingehalten worden? | Maßnahmen AM2 | |
| 2. Sind die vereinbarten Tools benutzt worden? | Maßnahmen AM2 | |
| 3. Stimmt der Code mit der Matrixspezifikation überein? | Schnittstellenbeschreibung AM1/Modulspezifikation BM1 | |
| Datum: | | |
| Name: | | |
| Softwaresignatur: | | |

## 6.14 Zusammenfassung der matrixbasierten Dokumentation

Die folgenden Punkte nennen die wesentlichen Ergebnisse dieses Kapitels:

- Eine pragmatische und transparente Vorgehensweise zur Abarbeitung des weiter vereinfachten V-Modells wurde angegeben. Sie soll einen roten Faden darstellen.
- Diese Vorgehensweise ist grundsätzlich unabhängig von der verwendeten Steuerung, vom vorgegebenen $PL_r$ und der verwendeten, geeigneten Programmiersprache.
- Die Kernpunkte der Vorgehensweise sind:
  - Aufteilung der Software in eine Vorverarbeitungsebene, Ansteuerungslogik und Ansteuerungsebene.
  - Beschreibung der Ansteuerungslogik durch eine C&E-Matrix bzw. durch eine transponierte und reduzierte C&E-Matrix. Letztere eignet sich insbesondere für größere Mengengerüste von Sicherheitsfunktionen.
  - Durch die Matrixdarstellungen werden die einzelnen Betriebsarten transparent.
  - Mit der Matrixdarstellung lässt sich das Anwendungsprogramm spezifizieren und es lässt sich nachvollziehbar verifizieren und validieren.
  - Zur Vervollständigung der Testabdeckung kann man zusätzliche Testfälle zu der Matrix hinzufügen.
  - Eine Codegenerierung mittels der Matrix ist prinzipiell möglich und würde das Arbeiten mit einer Sicherheits-SPS deutlich erleichtern.
- Die Softwarevalidierung besteht aus Analysen (der Verifikationen, des Codereviews) und Funktionstests (I/O-Liste, C&E-Matrix). Eine vollständige Testabdeckung ist in der Praxis nur schwer möglich. Man ist auf eine pragmatische Vorgehensweise angewiesen (Abschnitt 5.11).
- Selbst entwickelte einfache Funktionsbausteine (Bibliotheksbausteine) können mit ähnlichen Matrixmethoden spezifiziert und getestet werden. Bei komplexeren Funktionsbausteinen wird auf PLCopen [8] verwiesen.
- Die Spezifikation und die Testpläne für weitere herstellerspezifische Details (z. B. für Parameter von I/O-Karten, Parameter für Umrichter usw.) müssen noch zusätzlich erstellt und bearbeitet werden.

In Abbildung 36 werden die Zusammenhänge der Dokumente und Aktivitäten im V-Modell nochmals zusammengefasst dargestellt. Das Bild zeigt die Dokumente des V-Modells und ihre Zusammenhänge zu den überprüfenden Tätigkeiten. So wird die Hardware in den Dokumenten A2.3 und A2.4 überprüft und das Programm in den Dokumenten A2.4, A3, B1, B3 und B4. Dabei

steht die Markierung „A“ für die Analysen und die Markierung „T“ für Funktionstest. Mit „C1“ bzw. „D1“ an den Pfeilen ist auch direkt ersichtlich, ob es ein Teil des Codereviews bzw. der Softwarevalidierung ist. Zusätzlich kann man über den gestrichelten Pfeil sehen, dass die Dokumente B3 und B4 gegen die Spezifikation der Sicherheitsfunktionen (A1) verifiziert werden müssen.

Anmerkung: Die hier dargestellte Matrixmethode des IFA kann auch zur kompakten Darstellung von nicht sicherheitsrelevanten Schaltvorgängen genutzt werden.

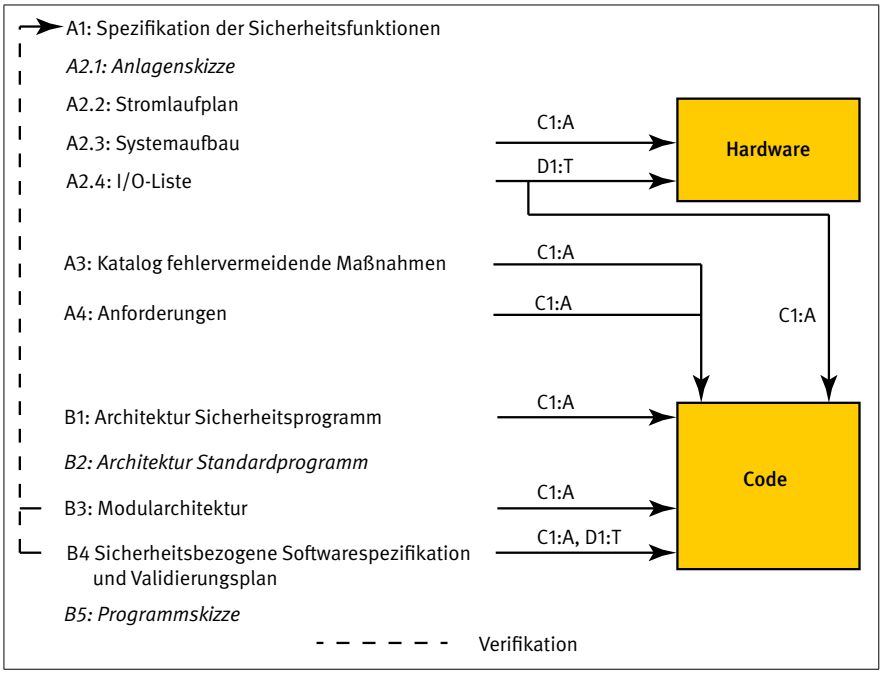



Abbildung 36:
Zusammenhang der Dokumente des V-Modells (kursiv = optional); A = Analyse, T = Test, C1 = Protokoll Codereview, D1 = Protokoll Softwarevalidierung

## 6.15 Verfahren für Modifikationen

Prozesssoftware muss z. B. bei Funktionsänderungen einer Maschine oder Anpassung der Hardware angepasst werden. Dies trifft für die Sicherheitssoftware genauso zu. So werden Sicherheitskonzepte verändert, zusätzliche Schutzeinrichtungen eingebaut oder auch nur der Einfluss einer Schutzeinrichtung auf den Arbeitsablauf der Maschine verändert.

Bei der Modifikation von Software muss in der Softwareentwicklung im V-Modell zurückgesprungen werden. Je tiefgreifender die Änderung ist, umso weiter vorne im V-Modell muss wieder angesetzt werden. Ein Verfahren zur Modifikation von Software soll hier an einem Beispiel gezeigt werden.

Das Änderungsmanagement besteht aus den folgenden Schritten:

1. Einflussanalyse: Eintragung der Änderung mit Klartextbeschreibung in das Formular „Änderungshistorie“ (Beispiel siehe Tabelle 18) inkl. der Nennung der betroffenen Dokumente.

2. Die Änderungen in den ausgewiesenen Dokumenten werden z. B. farblich gekennzeichnet.

3. Die Änderungen werden möglichst durch eine zweite Person verifiziert.

4. Codierung der Änderung.

5. Das Codereview führt möglichst eine zweite Person durch.

6. Die Änderungen werden validiert (I/O-Check bei neuen Signalen, Funktionstest).

7. Nach erfolgreicher Validierung werden die Farbmarkierungen wieder herausgenommen.

8. Archivieren der Versionen sämtlicher betroffener Dokumente.

Diese Vorgehensweise wird nun am Beispiel der Roboterzelle aus dem vorherigen Kapitel gezeigt. Als Änderung wird eine weitere Schutztür angebracht, die den Zugang für Wartungsarbeiten erleichtern soll (entspricht Beispiel in Abschnitt 7.3). Tabelle 18 zeigt die Eintragung in der Änderungshistorie mit der Einflussanalyse, welche Dokumente betroffen sind. Da mit einer weiteren Schutztür zwei neue Sicherheitsfunktionen integriert werden, muss das V-Modell für den geänderten Teil komplett abgearbeitet werden. Die Bezeichnungen der Dokumente entsprechen den Bezeichnungen aus Tabelle 1 zu den Dokumenten des V-Modells.

Tabelle 18:
Beispiel für eine Änderungshistorie (bezogen auf Beispiel Roboterzelle)

| Nr. | Datum | Änderung durch | Dokument | Änderung |
|---|---|---|---|---|
| 1 | 09.01.2013 | M. Muster | A1, A2.1, A2.2, A2.4, B3, B4, B5 | Schutztür 4 hinzugefügt für Wartungsarbeiten am Werkzeugträger |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |

Zur besseren Übersicht wird zunächst die geänderte Anlagenskizze (Dokument A2.1) in Abbildung 37 gezeigt. Neu aufgenommen wurde die Schutztür SG4.

Mit dieser neuen Schutztür gibt es zwei neue Sicherheitsfunktionen SF6 und SF7, die dafür sorgen, dass der Motor M2 hinter der Schutztür abschaltet, wenn die Schutztür geöffnet wird, und der Motor M1 abschaltet, wenn die Schutztüren SG4 und SG3 offen sind.

In Tabelle 19 sind die Sicherheitsfunktionen (Dokument A1) mit den neuen Sicherheitsfunktionen SF6 und SF7 (in roter Schrift) aufgelistet. Die Betriebsart wurde nicht angegeben, da es in diesem Beispiel nur die Betriebsart Automatikbetrieb gibt.

Im Schaltplan sind durch die neue Schutztür SG4 zwei weitere Eingänge zur Überwachung der Schutztür hinzugekommen. Abbildung 38 zeigt den geänderten Stromlaufplan mit farblich gekennzeichneten Kontakten (Dokument A2.2).

Mit diesen zwei neuen Eingängen ändert sich die I/O-Liste (Dokument A2.4). Sie sind in Tabelle 20 – in roter Schrift gekennzeichnet – aufgenommen.

Abbildung 37:
Anlagenskizze (Dokument A2.1) der Roboterzelle zusätzlicher Schutztür

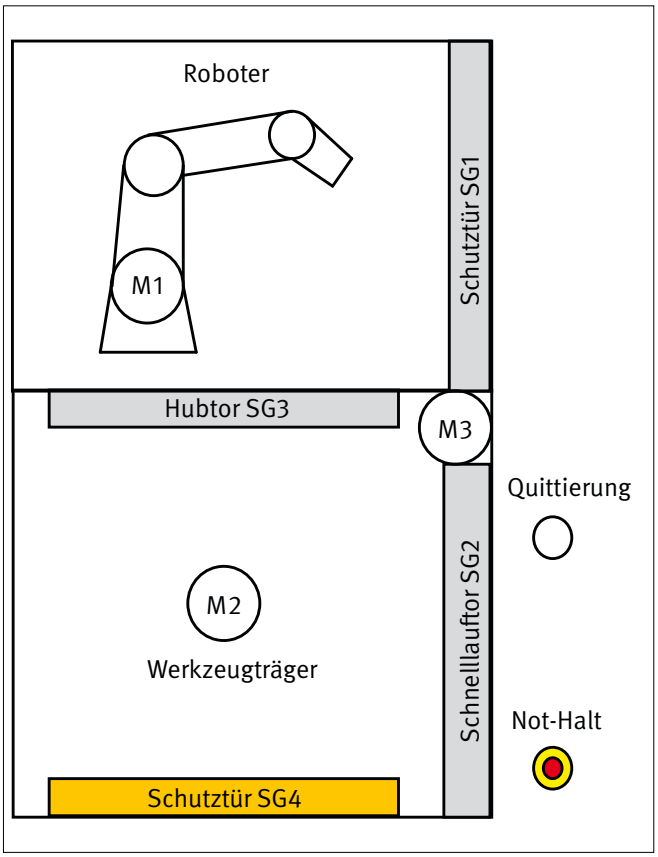



Tabelle 19:
Sicherheitsfunktionen (Dokument A1) der Roboterzelle mit zusätzlicher Schutztür

| Nr. | Beschreibung | $PL_r$ | Reaktionszeit in ms | Priorität |
|---|---|---|---|---|
| SF1 | Wenn der Not-Halt EMST betätigt wird, werden M1, M2 und M3 abgeschaltet. | d | 100 | 1 |
| SF2 | Wenn Schutztür SG1 auf, dann wird M1 abgeschaltet. | d | 100 | 2 |
| SF3 | Wenn Schnelllauftor SG2 auf, dann wird M2 abgeschaltet. | d | 100 | 2 |
| SF4 | Wenn Schnelllauftor SG2 auf und Hubtor SG3 auf, dann wird M1 abgeschaltet. | d | 100 | 2 |
| SF5 | Wenn Sicherheitsleiste SL_SG2 des Schnelllauftors SG2 betätigt wird, dann wird der Motor M3 abgeschaltet. | d | 100 | 2 |
| **SF6** | **Wenn Schutztür SG4 auf, dann wird M2 abgeschaltet.** | **d** | **100** | **2** |
| **SF7** | **Wenn Schutztür SG4 auf und Hubtor SG3 auf, dann wird M1 abgeschaltet.** | **d** | **100** | **2** |

Abbildung 38:
Stromlaufplan (Dokument A2.2) der Roboterzelle mit zusätzlicher Schutztür

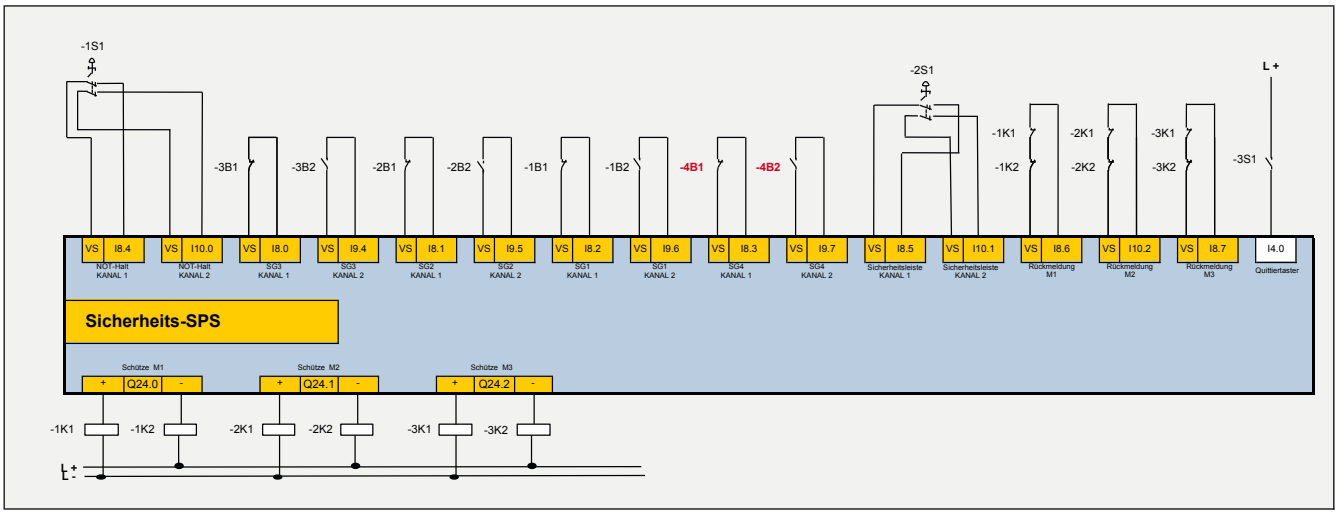



Tabelle 20:
I/O-Liste (Dokument A2.4) der Roboterzelle mit zusätzlicher Schutztür

| Eingänge | Variable | Adresse | D1 Validiert (ok/n. ok) | C1 richtige Verschaltung in der Software verifiziert (ok/n. ok) |
|---|---|---|---|---|
| **Eingänge** | | | | |
| Not-Halt, zweikanalig (NC) (1S1) | IS_EMST | %I8.4 | | |
| Kontakt 1 Schutztür Roboter SG1 (NC) (1B1) | IS_SG1_1 | %I8.2 | | |
| Kontakt 2 Schutztür Roboter SG1 (NO) (1B2) | IS_SG1_2 | %I9.6 | | |
| Kontakt 1 Schnelllauftor SG2 (NC) (2B1) | IS_SG2_1 | %I8.1 | | |
| Kontakt 2 Schnelllauftor SG2 (NO) (2B2) | IS_SG2_2 | %I9.5 | | |
| Kontakt 1 Hubtor SG3 (NC) (3B1) | IS_SG3_1 | %I8.0 | | |
| Kontakt 2 Hubtor SG3 (NO) (3B2) | IS_SG3_2 | %I9.4 | | |
| **Kontakt 1 Schutztür SG4 (NC) (4B1)** | **IS_SG4_1** | **%I8.3** | | |
| **Kontakt 2 Schutztür SG4 (NO) (4B2)** | **IS_SG4_2** | **%I9.7** | | |
| Sicherheitsleiste von SG2, zweikanalig (NC) (2S1) | IS_SL_SG2 | %I8.5 | | |
| Rückmeldung Schütze M1 (NC) (1K1, 1K2) | IS_SM1 | %I8.6 | | |
| Rückmeldung Schütze M2 (NC) (2K1, 2K2) | IS_SM2 | %I10.2 | | |
| Rückmeldung Schütze M3 (NC) (3K1, 3K2) | IS_SM3 | %I8.7 | | |
| Quittiertaster (NO) (3S1) | I_ACK | %I4.0 | | |
| **Ausgänge** | | | | |
| Schütze Motor M1 (1K1, 1K2) | QS_M1 | %Q24.0 | | |
| Schütze Motor M2 (2K1, 2K2) | QS_M2 | %Q24.1 | | |
| Schütze Motor M3 (3K1, 3K2) | QS_M3 | %Q24.2 | | |
| | | Datum: | | |
| | | Name: | | |
| | | Softwaresignatur: | | |

In der Modularchitektur (Dokument B3) wird der zusätzliche Schutztürbaustein mit aufgeführt. Abbildung 39 zeigt diese ergänzte Modularchitektur. Die C&E-Matrix (Dokument B4) ändert sich ebenfalls: Sie erhält in Abbildung 40 zwei weitere Zeilen für die hinzugekommenen Sicherheitsfunktionen.

Bei der kompakten Form der Matrixdarstellung (Abbildung 41) zeigen sich die Änderungen nur in zwei Zellen.

In der Programmskizze in Abbildung 42 (Dokument B5) ist die Änderung aufgrund der farblichen Abhebung gut zu erkennen.

Abbildung 39:
Modularchitektur (Dokument B3) der Roboterzelle mit zusätzlicher Schutztür

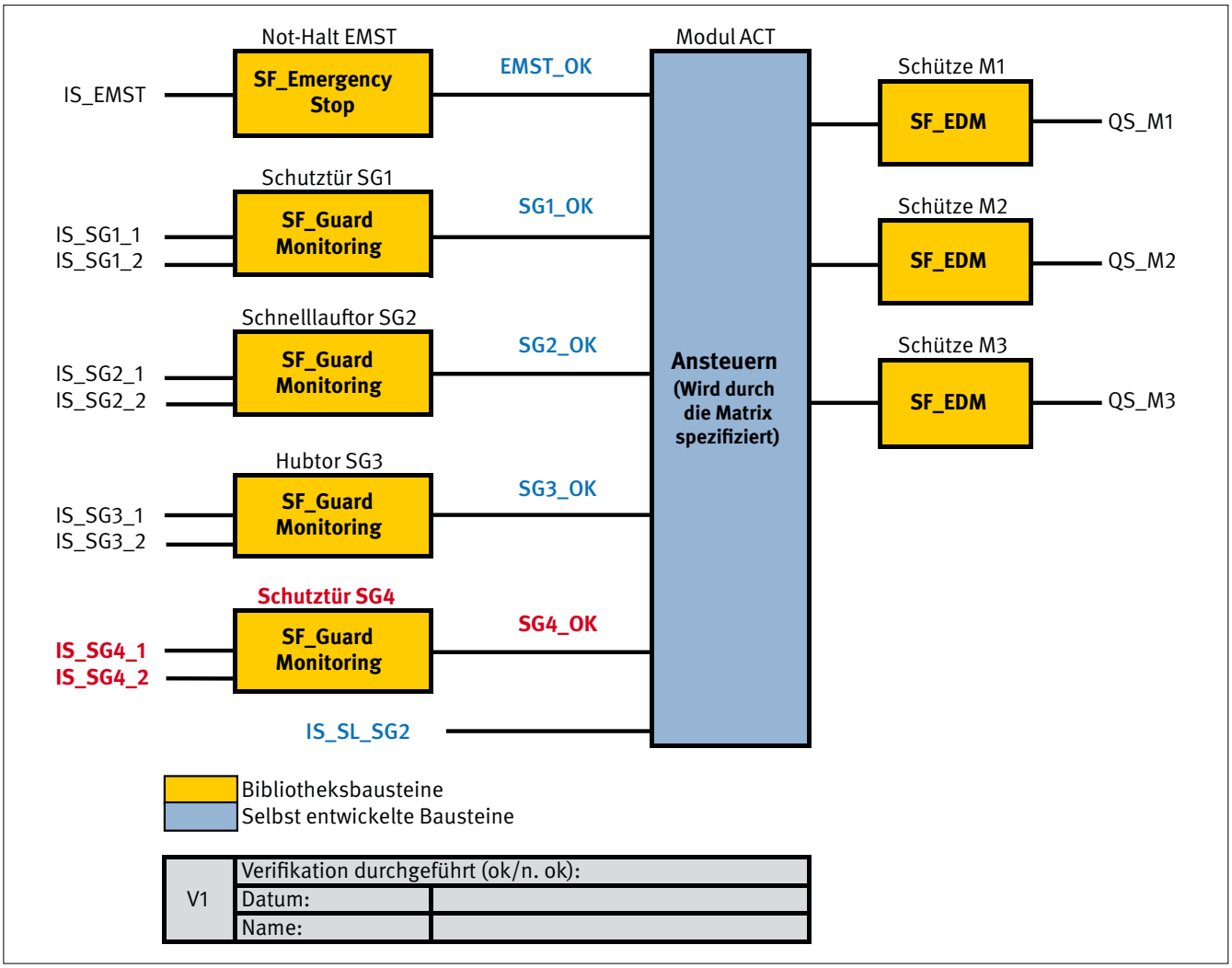


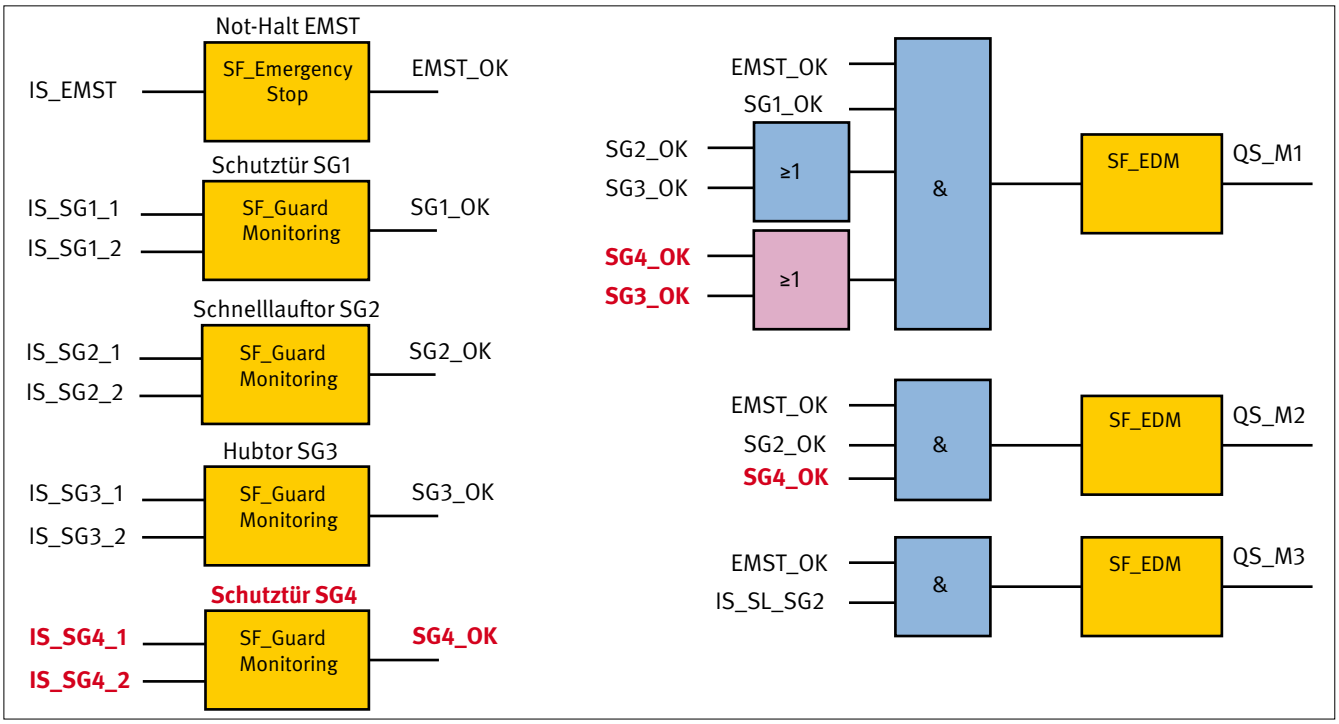

Abbildung 40:
C&E-Matrix (Dokument B4) der Roboterzelle mit zusätzlicher Schutztür

| | | | Cause | | | | | | | | | | | Effect | | | | D1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Schaltzustände der Eingänge | | | | | | | | | | | Ausgänge | | | | | | |
| Betriebsart | Vorgängerzustand bei Test | Zustand | IS_IMST (I8.4) | IS_SG1_1 (I8.2) | IS_SG1_2 (I9.6) | IS_SG2_1 (I8.1) | IS_SG2_2 (I9.5) | IS_SG3_1 (I8.0) | IS_SG3_2 (I9.4) | IS_SG4_1 (I8.2) | IS_SG4_2 (I9.7) | IS_SL_SG2 (I8.5) | Bezeichnung | QS_M1 (Q24.0) | QS_M2 (Q24.1) | QS_M3 (Q24.2) | Quittierung I_ACK (I4.0) | Geprüft (ok/n. ok) | Name | Datum |
| | | | | | | | | | | | | | C1: Software entspricht der Matrixdokumentation | | | | | | | |
| | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ALL_OK | EIN | EIN | EIN | | | | |
| | 1 | 2 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | SF1: Not-Halt EMST betätigt | EMST_OK AUS | EMST_OK AUS | EMST_OK AUS | EIN | | | |
| | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | SF2: SG1 offen | SG1_OK AUS | NOP | NOP | EIN | | | |
| | 1 | 4 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | SF3: SG2 offen | NOP | SG2_OK AUS | NOP | EIN | | | |
| | 1 | 5 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | SF4: SG2 und SG3 offen | SG2_OK v SG3_OK AUS | NOP | NOP | EIN | | | |
| | 1 | 6 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | SF5: Sicherheitsleiste betätigt | NOP | NOP | IS_SL_SG2 AUS | EIN | | | |
| | 1 | 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | SF6: SG4 offen | NOP | SG4_OK AUS | NOP | EIN | | | |
| | 1 | 8 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | SF7: SG4 und SG3 offen | SG4_OK v SG3_OK AUS | NOP | NOP | EIN | | | |
| V1 | Verifikation durchgeführt (ok/n. ok): | | | | | | | | | | | | | | | Softwaresignatur: | | | | |
| | Datum: | | | | | | | | | | | | | | | | | | | |
| | Name: | | | | | | | | | | | | | | | | | | | |

Abbildung 41:
Kompakte Matrixdarstellung der Roboterzelle mit zusätzlicher Schutztür

| Aktoren | | Abschaltungen | | C1 | | | D1 | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | beteiligte SFs | Software entspricht der Matrixdokumentation | | | Funktion validiert | | |
| Ausgang | Bezeichnung | Betriebsart alle | | ok/n. ok | Name | Datum | ok/n. ok | Name | Datum |
| QS_M1 | Motor M1 | EMST_OK & SG1_OK & (SG2_OK v SG3_OK) & (SG4_OK v SG3_OK) | 1, 2, 4, 7 | | | | | | |
| QS_M2 | Motor M2 | EMST_OK & SG1_OK & SG2_OK & SG4_OK | 1, 3, 6 | | | | | | |
| QS_M3 | Motor M3 | EMST_OK&IS_SL_SG2 | 1, 5 | | | | | | |
| V1 | Verifikation durchgeführt (ok/n. ok): | | | Softwaresignatur: | | | | | |
| | Datum: | | | | | | | | |
| | Name: | | | | | | | | |

Abbildung 42:
Programmskizze (Dokument B5) der Roboterzelle mit zusätzlicher Schutztür




Am Dokument des Codereviews (C1) hat sich nichts geändert. Allerdings muss das Codereview für die Änderungen neu durchgeführt werden. Ebenso müssen die geänderten Teile des Programms überprüft werden. Dazu sind die I/O-Liste (Dokument A2.4) für die neuen Signale sowie die neuen Zeilen der C&E-Matrix (Dokument B4) bzw. die geänderten Zeilen der kompakten Matrixdarstellung zu verifizieren und validieren.

Diese Vorgehensweise soll Änderungen möglichst transparent verfolgbar machen, damit es bei der Modifikation von Softwarekomponenten nicht zu Fehlern kommt. Wenn eine strukturierte Vorgehensweise zur Softwareentwicklung eingehalten wird, wie die beschriebene Matrixmethode des IFA, kann der Aufwand bei einer Änderung auf das oben Beschriebene reduziert werden. Ansonsten muss nach einer Änderung eventuell das komplette Programm neu überprüft werden.

## 6.16 Vereinfachung bei wiederkehrenden Sicherheitsfunktionen

Sicherheitsfunktionen wie Not-Halt kommen an Maschinen häufig vor und je größer eine Maschine ist, desto mehr Not-Halt-Schaltgeräte werden an ihr angebracht. Diese können mit einer Sicherheitsfunktion in der Software bearbeitet werden, allerdings ist es wichtig, jedes einzelne Not-Halt-Schaltgerät zu testen. Um eine Übersicht für die Tests zu schaffen, werden alle Not-Halt-Schaltgeräte in einer eigenen Tabelle zusammengefasst.

Prinzipiell gibt es zwei Möglichkeiten der Verschaltung: Zum einen können alle Not-Halt-Schaltgeräte hardwareseitig hintereinander in Reihe geschaltet werden und belegen dann einen Binäreingang pro Kanal in der SPS. Zum anderen können die Not-Halt-Schaltgeräte auf separate Eingänge der SPS verdrahtet

werden und die Reihenschaltung erfolgt softwareseitig mit einem UND-Glied in der SPS.

Die Diskrepanz der beiden Kanäle kann sowohl direkt in der SPS-Eingangskarte überwacht werden als auch mit einem Bibliotheksbaustein (z. B. SF_Equivalent nach PLCopen [8]) im Programm. Um die Überwachung der Diskrepanz der beiden Not-Halt-Kanäle zu gewährleisten, muss nicht jeder Kanal explizit getestet werden, sondern die richtige Parametrierung der Eingangskarten und mit dem I/O-Check die korrekte Verdrahtung überprüft werden.

Abbildung 43 zeigt den Not-Halt-Funktionstest, entnommen aus dem Beispiel der Rundtischanlage (Abschnitt 7.4). Bei diesem Beispiel sind an der Anlage 18 Not-Halt-Taster verbaut, die in Reihe geschaltet die Sicherheitsfunktion „Not-Halt“ anfordern.

Es sind die Betriebsmittelkennzeichen aller Not-Halt-Taster angegeben und dazu Validierungsspalten, in denen der Funktionstest jedes Not-Halt-Tasters bestätigt wird. Beim Betätigen jedes Not-Halt-Schaltgerätes ist zu überprüfen, ob sich der Pegel an beiden Eingängen der SPS verändert. So wird für jeden Not-Halt-Taster auch ein I/O-Check vorgenommen, mit dem die richtige Verdrahtung sichergestellt wird.

Werden die Not-Halt-Schaltgeräte nicht in Reihe geschaltet, sondern einzeln auf Eingänge gelegt, lässt sich die Tabelle in Abbildung 44 um eine Spalte mit der Angabe der Eingänge erweitern.

Abbildung 43:
Beispiel Not-Halt-Funktionstest

| **Not-Halt-Taster** I0.0 | **D1** **Funktion getestet** | | |
|---|---|---|---|
| | ok/n. ok | Name | Datum |
| -1S1 | | | |
| -1S2 | | | |
| -1S3 | | | |
| -1S4 | | | |
| -1S5 | | | |
| -1S6 | | | |
| -1S7 | | | |
| -1S8 | | | |
| -1S9 | | | |
| -1S10 | | | |
| -1S11 | | | |
| -1S12 | | | |
| -1S13 | | | |
| -1S14 | | | |
| -1S15 | | | |
| -1S16 | | | |
| -1S17 | | | |
| -1S18 | | | |

| **Not-Halt-Taster** | | **D1** **Funktion getestet** | | |
|---|---|---|---|---|
| **BMK** | **Eingänge** | ok/n. ok | Name | Datum |
| -1S1 | I0.0/I1.4 | | | |
| -1S2 | I0.1/I1.5 | | | |
| -1S3 | I0.2/I1.6 | | | |
| -1S4 | I0.3/I1.7 | | | |
| -1S5 | I0.4/I2.0 | | | |
| -1S6 | I0.5/I2.1 | | | |
| -1S7 | I0.6/I2.2 | | | |
| -1S8 | I0.7/I2.3 | | | |
| -1S9 | I1.0/I2.4 | | | |
| -1S10 | I1.1/I2.5 | | | |
| -1S11 | I1.2/I2.6 | | | |
| -1S12 | I1.3/I2.7 | | | |
| -1S13 | I10.0/I11.4 | | | |
| -1S14 | I10.1/I11.5 | | | |
| -1S15 | I10.2/I11.6 | | | |
| -1S16 | I10.3/I11.7 | | | |
| -1S17 | I10.4/I12.0 | | | |
| -1S18 | I10.5/I12.1 | | | |

Abbildung 44:
Beispiel Not-Halt-Funktionstest mit Eingangsangaben

Ob nun das Not-Halt-Signal in der Hardware zusammengefasst wird – durch die Reihenschaltung der Not-Halt Schaltgeräte – oder in der Software durch eine Reihenschaltung der Eingänge in der SPS, spielt für die Funktion keine Rolle.

Zur Fehlersuche in großen Anlagen eignet sich die Nutzung separater Eingänge, da der Fehlerort schneller zu finden ist.

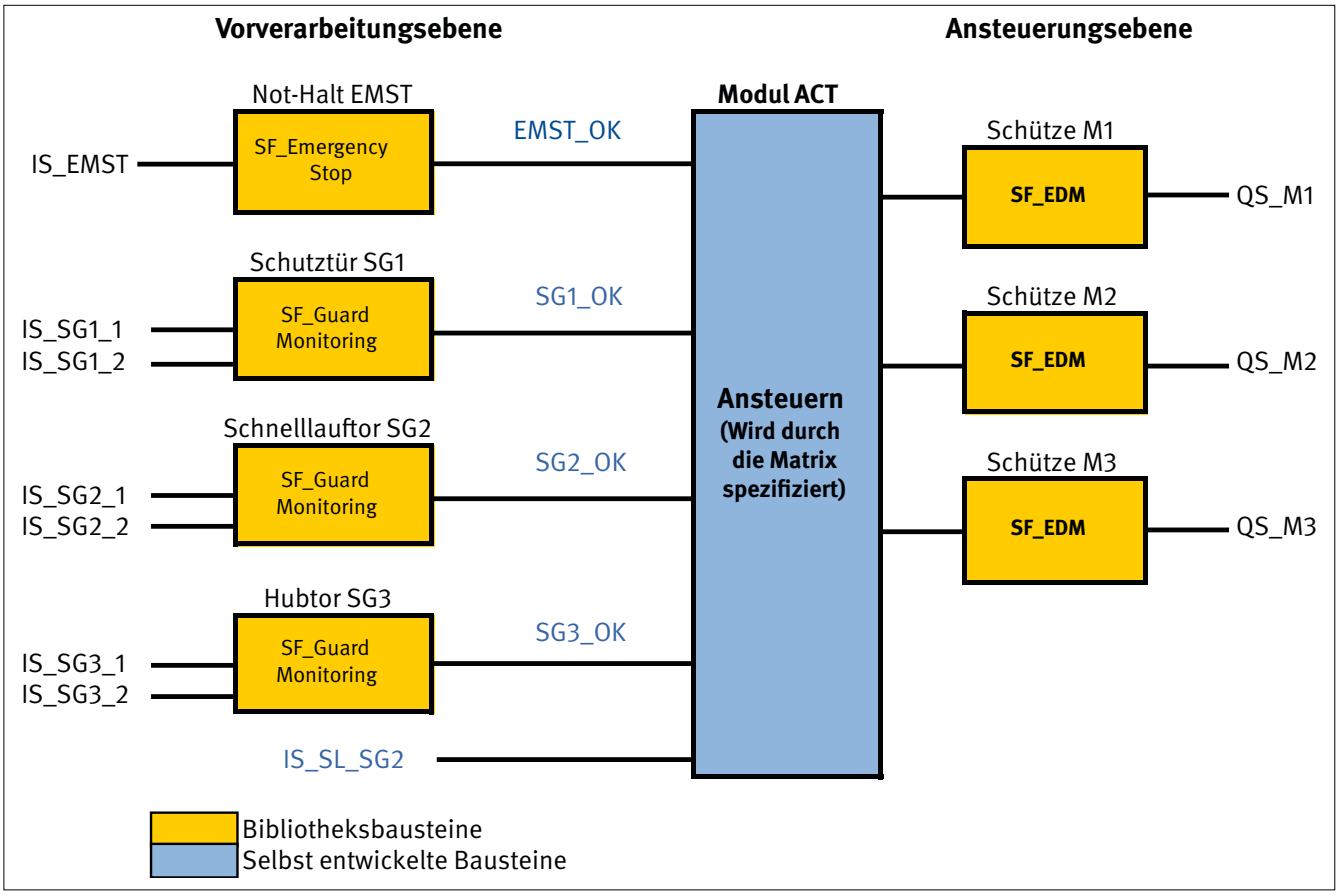
Mit diesen Tabellen wird sichergestellt, alle Not-Halt-Schaltgeräte getestet zu haben. Mit der Matrix wird die Sicherheitsfunktion getestet. Beim Test der Sicherheitsfunktion mit der Matrix muss dann nur ein Not-Halt-Schaltgerät betätigt werden.

## 6.17 Berücksichtigung von fehlerbeherrschenden Maßnahmen

Bei Steuerungen für Maschinen können zwei Ausfallarten unterschieden werden: Ausfälle aufgrund von systematischen Fehlern und von zufälligen Fehlern. Systematische Fehler sollen durch eine strukturierte und transparente Arbeitsweise vermieden werden. Zur Beherrschung von zufälligen Fehlern (aufgrund von Bauteildefekten, -alterung, -verschleiß) sind in Bibliotheksbausteinen Maßnahmen enthalten, die solche Fehler erkennen, dann das Ansteuersignal bzw. Ausgangssignal zurücknehmen und einen sicheren Zustand der Maschine einleiten.

Bei der gewählten Modularchitektur mit Vorverarbeitungsebene, Ansteuerlogik (Modul ACT) und Ansteuerungsebene (siehe Abbildung 45 mit Beispiel aus Abschnitt 6.2) werden in der Vorverarbeitungsebene und der Ansteuerungsebene Bibliotheksbausteine verwendet. Im Modul ACT werden keine fehlerbeherrschenden Maßnahmen realisiert.

Abbildung 45:
Beispiel Modularchitektur




Im Folgenden werden einige Beispiele für diese fehlerbeherrschenden Maßnahmen mit Funktionsbausteinen aus der PLCopen-Bibliothek [8] dargestellt. Im Einzelnen sind das die Funktionsbausteine SF_EDM zur Schützüberwachung, der SF_GuardMonitoring zur Schutztürüberwachung und der SF_EmergencyStop zur Not-Halt-Überwachung. Die Sicherheitsbausteine aus den Bibliotheken anderer SPS-Hersteller funktionieren ähnlich und haben ebenfalls Fehlererkennungen integriert.

Abbildung 46 zeigt den Baustein SF_EDM, wie er in der Software eingebunden wird. Dieser Baustein SF_EDM überwacht den Zustand des angesteuerten Schützes mit den Rückleseeingängen S_EDM1 und S_EDM2. Bei richtiger Funktion des Schützes besitzen diese Eingänge immer den inversen Zustand zum Ausgang S_EDM_Out. Ist dies nicht der Fall, so wird ein Rücklesefehler nach dem Ablauf des Zeitparameters MonitoringTime erkannt und gespeichert. Bei aktivem Rücklesefehler kann der Ausgang S_EDM_Out nicht mehr auf TRUE geschaltet werden, also z. B. das Schütz nicht eingeschaltet werden.

In Abbildung 47 ist der Aufruf des Bausteins SF_GuardMonitoring zu sehen. Der SF_GuardMonitoring überwacht eine Schutztür. Sobald einer der beiden Schutztürkontakte (Eingänge S_GuardSwitch1,2) den Zustand FALSE annimmt, wird der Ausgang S_GuardMonitoring ebenfalls auf FALSE gesetzt. S_StartReset steuert das Quittierverhalten. Automatisches Quittieren darf nur aktiviert werden, wenn systembedingt automatisches Anlaufen der Maschine nicht möglich ist.

Als letztes Beispiel zeigt Abbildung 48 den Aufruf des SF_EmergencyStop-Bausteins, der Not-Halt überwacht. Mit dem Baustein SF_EmergencyStop können Not-Halt-Abschaltungen mit Stopp-Kategorie 0 realisiert werden. Der Ausgang S_EStopOut wird auf FALSE gesetzt, sobald am Eingang S_EstopIn ein FALSE-Signal anliegt. Die Ausgänge werden erst wieder nach einer Quittierung

aktiviert. Mit S_AutoReset darf nur eine automatische Quittierung eingestellt werden, wenn automatisches Anlaufen der Maschine anderweitig verhindert wird.

Derartige fehlerbeherrschende Maßnahmen sind in vielen Sicherheits-Bibliotheksbausteinen von Steuerungsherstellern enthalten. Auch aus diesem Grund wird in den Programmierregeln die Verwendung von Bibliotheksbausteinen vorgeschrieben. Ebenfalls darf in der Regel kein automatischer Wiederanlauf stattfinden, um unvorhergesehene und eventuell gefahrbringende Aktionen einer Maschine zu unterbinden.

Abbildung 46:
Aufrufansicht des Bibliotheksbausteines SF_EDM

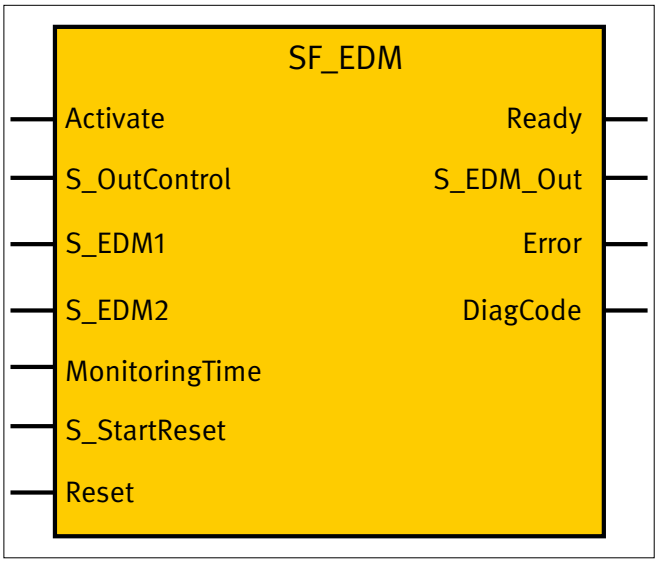



Abbildung 47:
Aufrufansicht des Bibliotheksbausteines SF_GuardMonitoring

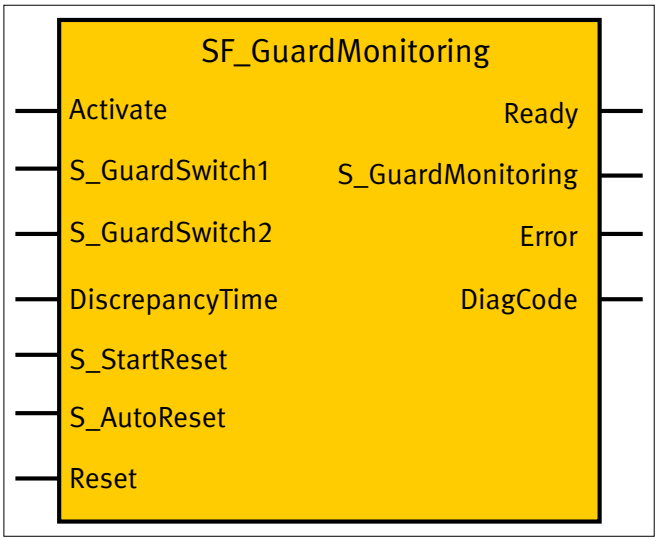
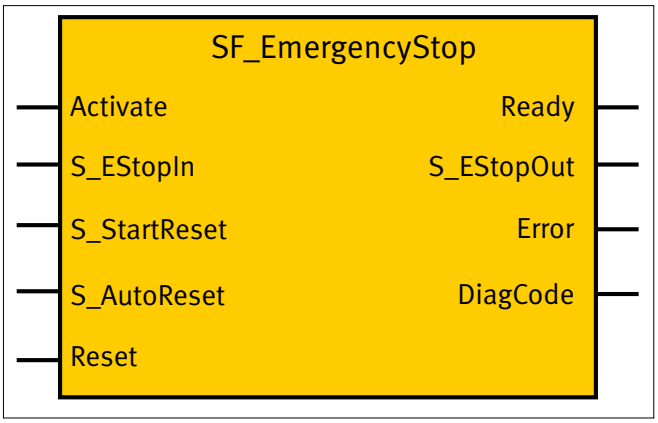



Abbildung 48:
Aufrufansicht des Bibliotheksbausteines SF_EmergencyStop



# 7 Übersicht über die behandelten Softwarebeispiele

In diesem Report wurde zunächst allgemein auf die Gestaltung sicherheitsbezogener Anwendungssoftware (SRASW) eingegangen. Die Matrixmethode des IFA ist zwar Schritt für Schritt an einem Beispiel beschrieben, aber einige dieser Schritte erfordern Übung. Sie lassen sich aufgrund der Vielfalt möglicher Sicherheitsfunktionen und ihrer Realisierung auch nur schwer allgemein beschreiben. Daher wird in diesem Kapitel die Dokumentation einer Vielzahl von Softwarebeispielen vorgestellt, die typische Sicherheitsfunktionen realisieren. Alle Beispiele sind nach der Matrixmethode dokumentiert und zum Download verfügbar.

Die Schaltbilder sind Prinzip-Schaltbilder, die sich ausschließlich darauf beschränken, die Sicherheitsfunktion(en) mit den hierzu notwendigen relevanten Komponenten zu zeigen. Nicht dargestellt werden zwecks besserer Übersicht solche schaltungstechnischen Maßnahmen, die in der Regel immer zusätzlich realisiert sein müssen, um z. B. den Berührungsschutz sicherzustellen, Über- und Unterspannungen bzw. Überdruck/Unterdruck zu beherrschen, Isolationsfehler, Erd- und Kurzschlüsse z. B. auf extern verlegten Leitungen aufzudecken oder die erforderliche Störfestigkeit gegen elektromagnetische Einwirkungen zu garantieren. Dazu gehören in der Elektrik auch Schutzbeschaltungen wie Sicherungen und Dioden, z. B. als Freilaufdioden oder Entkopplungsdioden. Selbstverständlich muss gemäß den Fehlerlisten aus DIN EN ISO 13849-2 z. B. auch der Einfluss von Leitungskurzschlüssen im Zusammenhang mit der jeweiligen Sicherheitsfunktion und abhängig von den Einsatzbedingungen berücksichtigt werden. So müssen grundsätzlich alle verwendeten Bauteile entsprechend ihrer Spezifikation geeignet ausgewählt sein, Überdimensionierung gehört zu den bewährten Sicherheitsprinzipien.

Die Steuerungsbeispiele mit den Schaltbildern und erläuternden Texten beschränken sich also auf wesentliche Gesichtspunkte und sollen deshalb nicht als Anregung für eine Realisierung verstanden werden. Alle Dokumente für die Schaltungsbeispiele wurden mithilfe des Tools SOFTEMA (siehe Kapitel 14) in der zum Zeitpunkt der Erstellung dieses Reportes verfügbaren Prototypen-Version ausgeführt. Daher weichen sie von den Beispieldokumenten zum Forschungsbericht des DGUV Projektes FF-FP0319 (siehe Abschnitt 2.2) ab.

Die folgenden Abschnitte beschreiben kurz die wesentlichen Funktionen der Beispiele. Weitere Informationen sind in den dazugehörigen Microsoft-Excel-Dokumenten enthalten. Die Steuerungsprogramme selbst sind im Rahmen des DGUV Projektes FF-FP0319 von den Projektausführenden mit Siemens SIMATIC STEP7 V5.5 und Distributed Safety V5.4 erstellt – mit Ausnahme des letzten Beispiels. Dieses Beispiel ist mit dem Pilz PNOZmulti Configurator 9.0.1 erstellt.

Das Passwort für die Programmlistings der Siemens-Beispiele lautet: pw_fcpu. Das Passwort der Ebene 1 für das Beispiel mit PNOZmulti lautet: 1.

Die Excel-Dokumente sind in einer einzigen Archivdatei im Downloadbereich dieses IFA Reports zu finden. Die Excel-Dokumente können mit SOFTEMA (Kapitel 14) geöffnet und betrachtet werden. Die Programmdaten sind ausschließlich auf der Internetseite des Projektes FF-FP0319 [5] verfügbar.

Tabelle 21 zeigt die behandelten Softwarebeispiele mit den zugehörigen Excel-Dateien und den Programmlistings.

Die Excel-Arbeitsmappen sind in verschiedene Arbeitsblätter aufgeteilt. Tabelle 22 gibt eine Übersicht über die verschiedenen Arbeitsblätter.

Tabelle 21:
Übersicht der behandelten Softwarebeispiele

| Abschnitt | Softwarebeispiel | Excel-Dokument (.xlsx) (auf Reportseite) | Programmlisting (.zip) (auf Projektseite) |
|---|---|---|---|
| 7.1 | Roboterfertigungszelle | Roboterzelle | Roboterzelle |
| 7.2 | Roboterfertigungszelle mit Einrichtbetrieb | Roboterzelle Einrichtbetrieb | Roboterzelle_Einrichten |
| 7.3 | Roboterfertigungszelle mit zusätzlicher Schutztür | Roboterzelle mit Änderung | Roboterzelle_Aenderung |
| 7.4 | Rundtischanlage | Rundtischanlage | Rundtischanlage |
| 7.5 | Werkzeugmaschine | Werkzeugmaschine | Werkzeugmaschine |
| 7.6 | Safely-Limited Speed (SLS) mit Standard FU | SLS mit Standard FU | SLS_StandardFU |
| 7.7 | Safely-Limited Speed (SLS) mit Sicherheits FU | SLS mit Sicherheits FU | SLS_SafeFU |
| 7.8 | Muting | Muting | Muting |
| 7.9 | Zweihandbedienung | Zweihandbedienung | Zweihandbedienung |
| 7.10 | PNOZmulti | PNOZmulti | PNOZmulti |

Tabelle 22:
Übersicht der Arbeitsblätter der Excel-Dokumente

| Verwendete Arbeitsblätter |
|---|
| V-Modell |
| A1 Sicherheitsfunktionen |
| A2.1 Anlagenskizze |
| A2.2 Stromlaufplan |
| A2.3 Systemaufbau |
| A2.4 I/O-Liste |
| A3 Maßnahmen |
| A4 Anforderungen |
| B1 Architektur Sicherheitsprogramm |
| B2 Architektur Standardprogramm |
| B3 Modularchitektur |
| B4 Matrix C&E |
| B4 Matrix kompakt |
| B5 Programmskizze |
| C1 Codereview |
| D1 Validierung |
| Änderungen |
| Personen |
| Projekt |
| Dokumente |
| Protokoll |

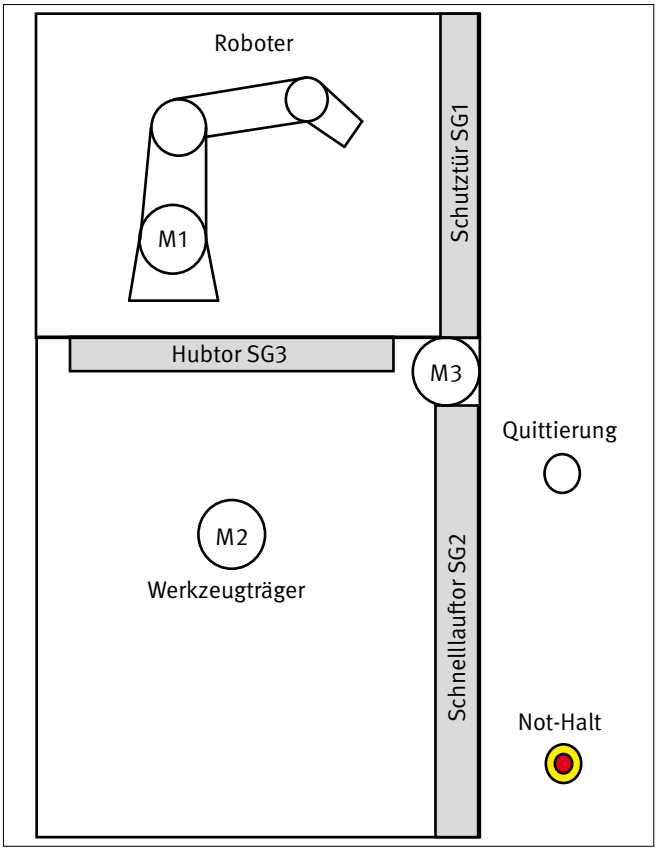
Das erste Arbeitsblatt „V-Modell“ zeigt als Übersicht die beiden V-Modelle für Software- und Modulentwicklung mit den zugehörigen Dokumenten. Die folgenden Arbeitsblätter tragen die Bezeichnungen der Dokumente des V-Modells. In Abschnitt 5.5 ist auch der jeweilige Inhalt beschrieben. Das Arbeitsblatt „Änderungen“ dient der Nachverfolgbarkeit von Änderungen des Anwendungsprogramms. Das Vorgehen ist in Abschnitt 6.15 beschrieben. Im Arbeitsblatt „Personen“ können die Personen dokumentiert werden, die an der Entwicklung, Validierung und Prüfung beteiligt sind. Das Arbeitsblatt „Projekt“ enthält alle projektbezogenen Informationen des Anwendungsprogramms. Im Arbeitsblatt „Dokumente“ können projektbezogene Dokumente und Dateien verwaltet werden. Im Arbeitsblatt „Protokoll“ werden sicherheitsrelevante Änderungen (z. B. beim $PL_r$ der Sicherheitsfunktionen) dokumentiert.

## 7.1 Roboterfertigungszelle

Im Beispiel Roboterfertigungszelle wird eine einfache Fertigungszelle, wie sie Abbildung 49 zeigt, behandelt. Es gibt nur die Betriebsart „Automatikbetrieb“. Das Beispiel wird auch in Abschnitt 6.2 behandelt.

In der Anlage wird das Werkzeug, das auf dem Werkzeugträger M2 montiert ist, von einem Roboter M1 mit Schaum befüllt. Danach zieht sich der Roboter wieder in seinen Bereich zurück. Das Hubtor SG3 ist ein automatisches Tor, das sich nur öffnen soll, wenn der Roboter das Werkzeug bestückt. Nachdem der Schaum ausgehärtet ist, öffnet eine Bedienperson das Schnelllauftor SG2 und entnimmt das fertige Werkstück aus dem Werkzeug. Danach schließt die Bedienperson das Schnelllauftor wieder. Am Schnelllauftor SG2 befindet sich eine Sicherheitsleiste SL_SG2 (nicht dargestellt), um Quetschungen des Bedienpersonals zu verhindern. Beim Auslösen der Sicherheitsleiste wird der Motor M3 des Schnelllauftors sofort abgeschaltet. Während das Schnelllauftor geöffnet ist, muss der Werkzeugträger stillgesetzt werden. Sollte das Hubtor nicht geschlossen sein, wenn das Schnelllauftor geöffnet wird, muss auch der Roboter aus Sicherheitsgründen stillgesetzt werden. Die Schutztür SG1 dient als Zugang zum Roboter für Wartungsarbeiten. Während die Schutztür offen ist, muss der Roboter stillgesetzt sein. Wird der Not-Halt EMST betätigt, werden der Roboter, der Werkzeugträger und das Schnelllauftor stillgesetzt.

Abbildung 49:
Anlagenskizze der Roboterfertigungszelle

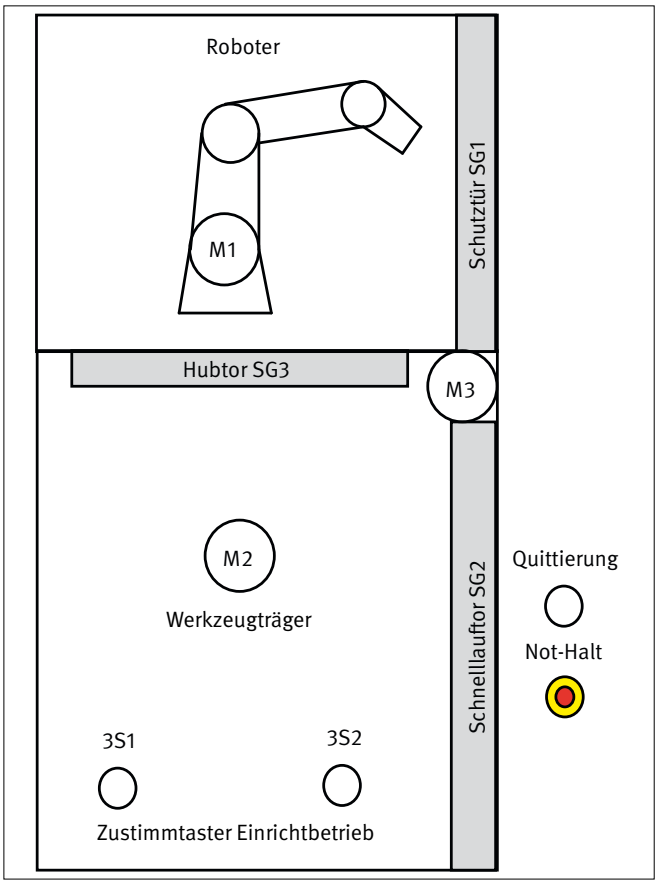


## 7.2 Roboterfertigungszelle mit Einrichtbetrieb

Abbildung 50 zeigt die Anlagenskizze des Beispiels Roboterfertigungszelle mit Einrichtbetrieb. Es handelt sich um eine Erweiterung des Beispiels 7.1. In diesem Beispiel gibt es die Betriebsarten „Automatikbetrieb“ und „Einrichtbetrieb“. Das Beispiel wird auch in Abschnitt 6.11 behandelt.

Die Funktion für den Automatikbetrieb entspricht dem Beispiel in Abschnitt 7.1. Wenn das Schnelllauftor SG2 geöffnet und das Hubtor SG3 geschlossen sind, kann im Einrichtbetrieb der Werkzeugträger M2 durch Betätigen eines der Zustimmtaster 3S1, 3S2 mit sicher begrenzter Geschwindigkeit (SLS) verfahren werden.

Abbildung 50:
Anlagenskizze der Roboterfertigungszelle mit Einrichtbetrieb

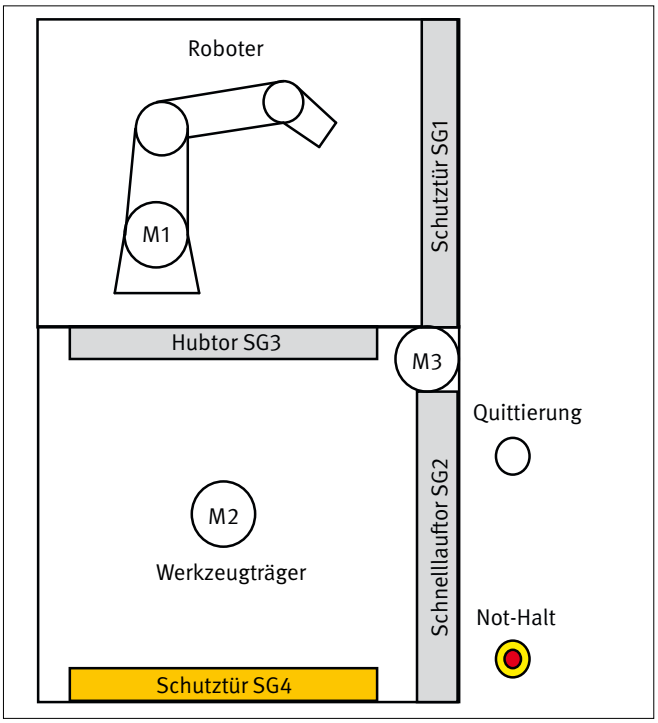



## 7.3 Roboterfertigungszelle mit zusätzlicher Schutztür

Die im Abbildung 51 gezeigte Anlagenskizze der Roboterfertigungszelle mit zusätzlicher Schutztür ist eine Erweiterung des Beispiels 7.1 um die Schutztür SG4. Es gibt nur die Betriebsart „Automatikbetrieb“. Das Beispiel wird auch in Abschnitt 6.15 behandelt.

Es ist die gleiche Funktion wie in Beispiel 7.1 umgesetzt. Zusätzlich wird der Werkzeugträger M2 stillgesetzt, wenn die Schutztür SG4 geöffnet wird. Der Roboter M1 muss stillgesetzt werden, wenn bei geöffneter Schutztür SG4 auch das Hubtor SG3 offen ist.

Abbildung 51:
Anlagenskizze der Roboterfertigungszelle mit zusätzlicher Schutztür

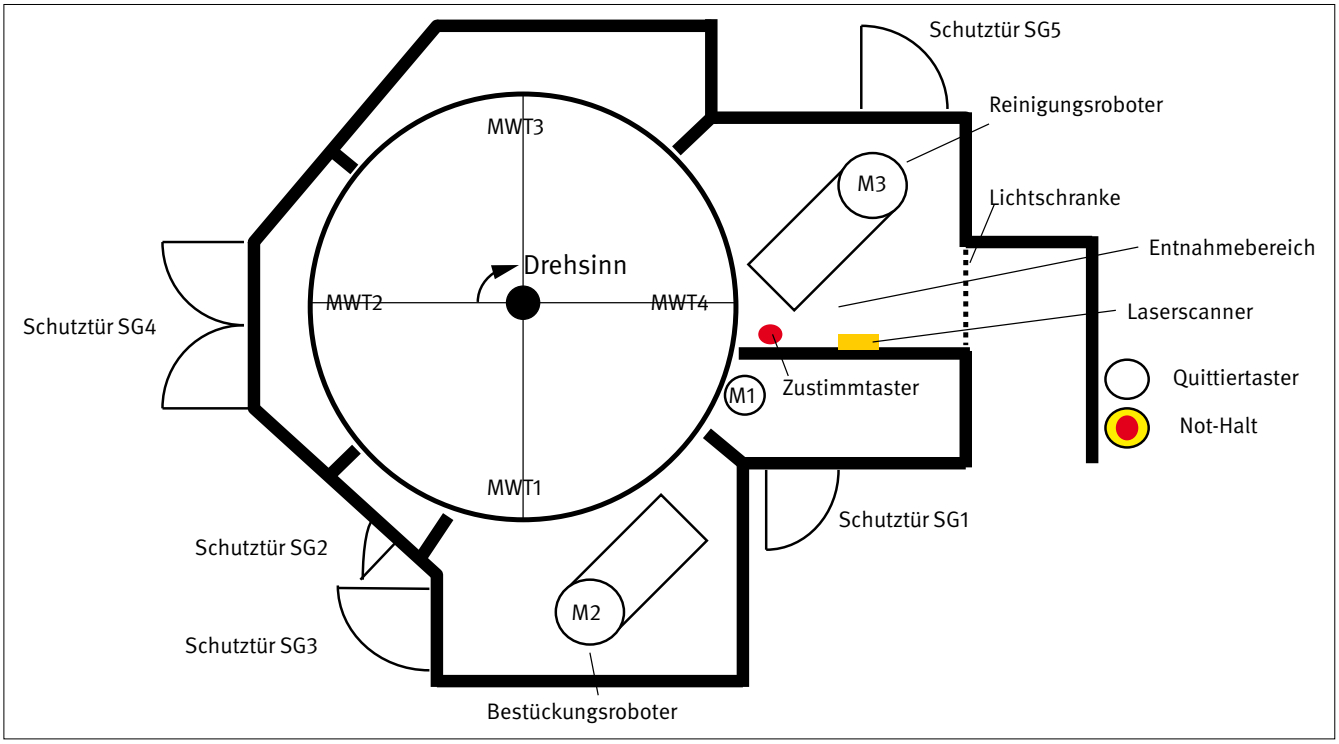


## 7.4 Rundtischanlage

Abbildung 52 stellt die Anlagenskizze des Beispiels Rundtischanlage dar. In diesem Beispiel gibt es die Betriebsarten „Automatikbetrieb“ und „Einrichtbetrieb“.

Bei dieser Anlage werden im Taktbetrieb Werkstücke hergestellt. Der Bestückungsroboter M2 befüllt einen Werkstückträger. Danach dreht der Rundtisch einen Takt weiter und der nächste Werkstückträger wird bestückt. So hat jedes Werkstück zwei Positionen zum Aushärten. In der vierten Position entnimmt eine Bedienperson das Werkstück. Danach reinigt ein Roboter M3 den Werkstückträger. Der Entnahmebereich wird durch eine Lichtschranke LSR überwacht und zusätzlich erkennt ein Laserscanner LS Personen im Entnahmebereich. Die Werkstückträger müssen alle mit Formen für Werkstücke oder mit Schutzgittern zur Überbrückung der Sicherheitskontakte bestückt sein. Fehlt eine Form oder das entsprechende Schutzgitter, werden alle vier Werkstückträgermotoren MWT1 bis MWT4 und der Drehtischmotor M1 abgeschaltet sowie der Bestückungsroboter und der Reinigungsroboter stillgesetzt. Beim Öffnen einer der Schutztüren SG1, SG2, SG4, SG5 oder Betätigen des Not-Halt EMST werden alle Werkstückträgermotoren abgeschaltet sowie der Drehtisch, der Bestückungsroboter und der Reinigungsroboter stillgesetzt. Wenn die Lichtschranke unterbrochen wird und sich kein Werkstückträger im Entnahmebereich befindet, schalten ebenfalls alle Motoren ab. Beim Öffnen der Schutztür SG3 wird der Bestückungsroboter stillgesetzt. Wenn die Achse 1 des Bestückungsroboters in eine Begrenzung fährt, wird der Bestückungsroboter stillgesetzt. Fährt der Reinigungsroboter in der Achse 1 in eine Begrenzung, wird er stillgesetzt. Befindet sich ein Werkstückträger im Entnahmebereich und die Lichtschranke wird unterbrochen, so schaltet der Motor des entsprechenden Werkstückträgers ab. Befinden sich die Achsen 1 und 2 des Reinigungsroboters nicht in der Grundposition, wenn die Lichtschranke unterbrochen wird oder der Laserscanner anspricht, wird der Reinigungsroboter stillgesetzt. Wenn der Reinigungsroboter sich in Grundposition befindet und der Laserscanner anspricht, kann der Drehtisch mit dem Zustimmtaster mit sicher begrenzter Geschwindigkeit (SLS) verfahren werden.

Abbildung 52:
Anlagenskizze der Rundtischanlage



## 7.5 Werkzeugmaschine

Im Beispiel „Werkzeugmaschine“ (Abbildung 53) wird eine Mehrachsfräsmaschine betrachtet. Diese Maschine hat die Betriebsarten „Automatikbetrieb“ und „Einrichtbetrieb“. Das Beispiel ist angelehnt an ein Beispiel aus [19].

Bei dieser Maschine gibt es einen beweglichen Werkstücktisch. Er kann mit dem Motor X1 in X-Richtung verfahren werden und ist mit dem Motor C drehbar gelagert. Das Fräswerkzeug ist in alle räumlichen Koordinaten beweglich mit den Achsmotoren X2, Y und Z. Die Werkzeugdrehung geschieht mit dem Spindelmotor S. Der automatische Werkzeugwechsler wird mit der Drehachse W und der pneumatischen Achse X3 gesteuert. Die Achsen X1, C, X2, Y, Z, S und W werden mit Antrieben mit integrierten Sicherheitsfunktionen gesteuert.

Im Automatikbetrieb müssen die Schutztüren geschlossen und zugehalten sein. Alle Schutztüren sind mit einer elektromechanischen Zuhaltung ausgerüstet. Wird der Not-Halt EMST betätigt, werden alle Antriebe mit SS1 stillgesetzt und die Energiezufuhr zur pneumatischen Achse X3 getrennt. SS1 steht für Safe Stop 1, d. h. der Antrieb wird an einer Rampe gebremst und dann mit STO (Safe Torque Off) von der Energiezufuhr getrennt.

Zum Öffnen der Schutztüren SG1 oder SG2 werden alle Antriebe mit SS2 heruntergefahren und die Energiezufuhr zu X3 getrennt. SS2 steht für Safe Stop 2, d. h. der Antrieb wird an einer Rampe gebremst und dann wird der sichere Stillstand mit SOS (Safe Operating Stop) überwacht. Diese Reaktion erfolgt auch beim Umschalten in den Einrichtbetrieb.

Zum Öffnen der Schutztür SG3 fahren die Antriebe des Werkzeugs (X2, Y, Z, S) und der Werkzeugwechsler W mit SS2 herunter und die Achse X3 wird stillgesetzt. Die Achsen des Werkstücktisches bleiben aktiv, da sie von der Schutztür SG3 aus nicht erreicht werden können.

Im Einrichtbetrieb kann das Öffnen der Schutztüren über Taster angefordert werden. Zur Sicherstellung des Stillstandes der pneumatischen Achse X3 wird die Zuhaltung der jeweiligen Schutztür erst nach fünf Sekunden entsperrt. Außerdem müssen die anderen Achsen den sicheren Stillstand über SOS melden. Bei der Schutztür SG3 wird der Stillstand der Achsen X1 und C nicht mit abgefragt.

Ist die Schutztür SG1 geöffnet und die anderen Schutztüren sind geschlossen, dann können im Einrichtbetrieb die Achsen des Werkstücktischs (X1, C) und die Bewegungsachsen des Werkzeugs (X2, Y, Z) mit sicher begrenzter Geschwindigkeit (SLS) mit den Tasten „Tippen_vorwärts“ oder „Tippen_rückwärts“ verfahren werden. Die Auswahl der Achsen geschieht über Anwahltaster. Bei geöffneter Schutztür SG2 und geschlossenen Schutztüren SG1 und SG3 kann im Einrichtbetrieb die Drehachse des Werkzeugwechslers W mit sicher begrenzter Geschwindigkeit (SLS) mit den Tasten „Magazin_vorwärts“ oder „Magazin_rückwärts“ verfahren werden. Dies geschieht zur Beladung des Werkzeugwechslers.

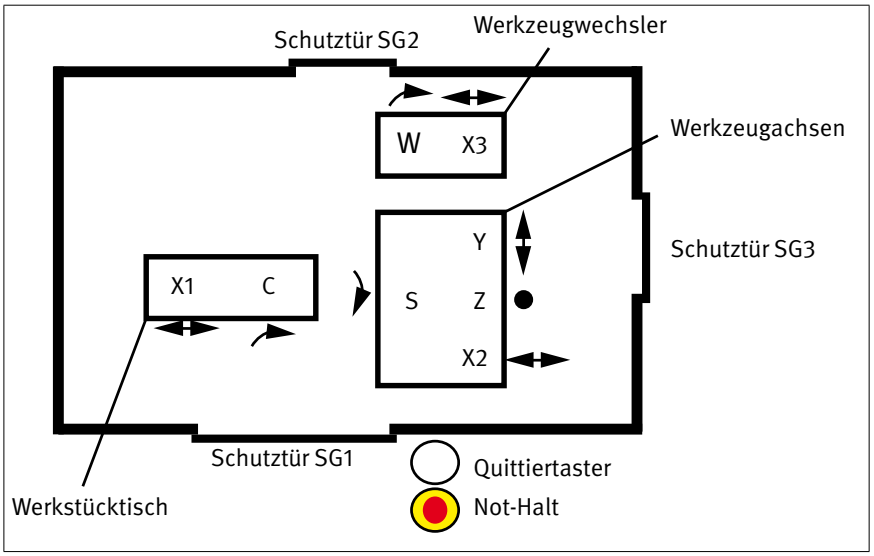



Abbildung 53:
Anlagenskizze einer Werkzeugmaschine

## 7.6 Safely-Limited Speed (SLS) mit Standard FU

Das Beispiel „SLS mit Standard FU“ zeigt den Aufbau einer Antriebssteuerung mit Sicherheits-SPS (Abbildung 54). Da der Umrichter keine Sicherheitsfunktionen integriert hat, muss die Drehzahl mit zwei Drehgebern G1 und G2 eingelesen und miteinander verglichen werden. Mit dieser zweikanaligen Ausführung kann die Sicherheitsfunktion realisiert werden.

Im Beispiel gibt es die Betriebsarten „Automatikbetrieb“ und „Einrichtbetrieb“. Im Automatikbetrieb darf der Motor bei geschlossener Schutztür drehen. Im Einrichtbetrieb darf der Motor bei geöffneter Schutztür über den Tipptaster S1 mit sicher begrenzter Geschwindigkeit (SLS) bewegt werden. Wenn der Not-Halt betätigt wird, wird der Motor in jedem Fall abgeschaltet.

Dieses Beispiel ist dem BGIA-Report 2/2008 [2] (Beispiel 21) entnommen.

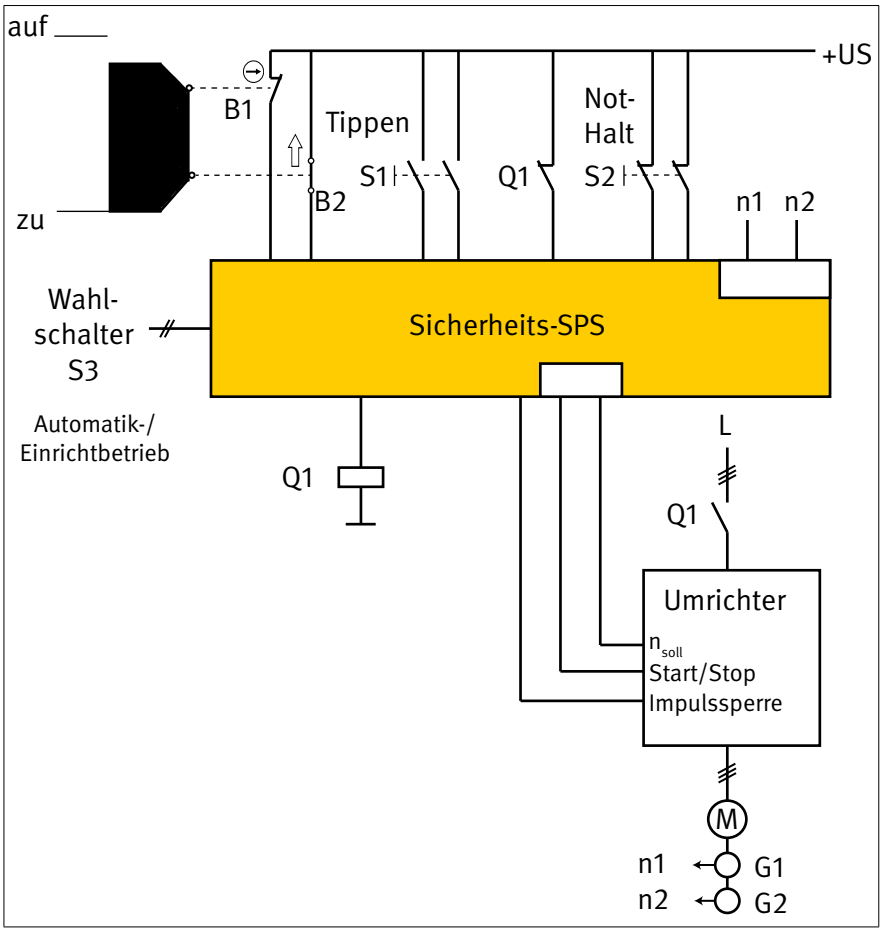



Abbildung 54:
Hardwareaufbau des Beispiels „SLS mit Standard FU“

## 7.7 Safely-Limited Speed (SLS) mit Sicherheits FU

Abbildung 55 zeigt den Hardwareaufbau des Beispiels „SLS mit Sicherheits FU“. Das Beispiel dient dem Vergleich des Aufwandes bei Nutzung von Antriebsreglern mit integrierten Sicherheitsfunktionen zu Antriebsreglern ohne Sicherheitsfunktionen wie im Beispiel 7.6. In diesem Beispiel werden die Sicherheitsfunktionen des Antriebsreglers von der Sicherheits-SPS angesteuert.

Die Sicherheitsfunktionen sind die gleichen wie im Beispiel in Abschnitt 7.6: Not-Halt, Schutztürüberwachung und SLS im Einrichtbetrieb.

Beim Betätigen des Not-Halts wird der Motor mit der Sicherheitsfunktion SS1 gestoppt. Wird die Schutztür geöffnet, wird der Motor mit SS2 gestoppt und dadurch mit SOS der Stillstand automatisch überwacht. Aus diesem Stillstand heraus kann mit dem Tipptaster die sicher begrenzte Geschwindigkeit (SLS) aktiviert werden.

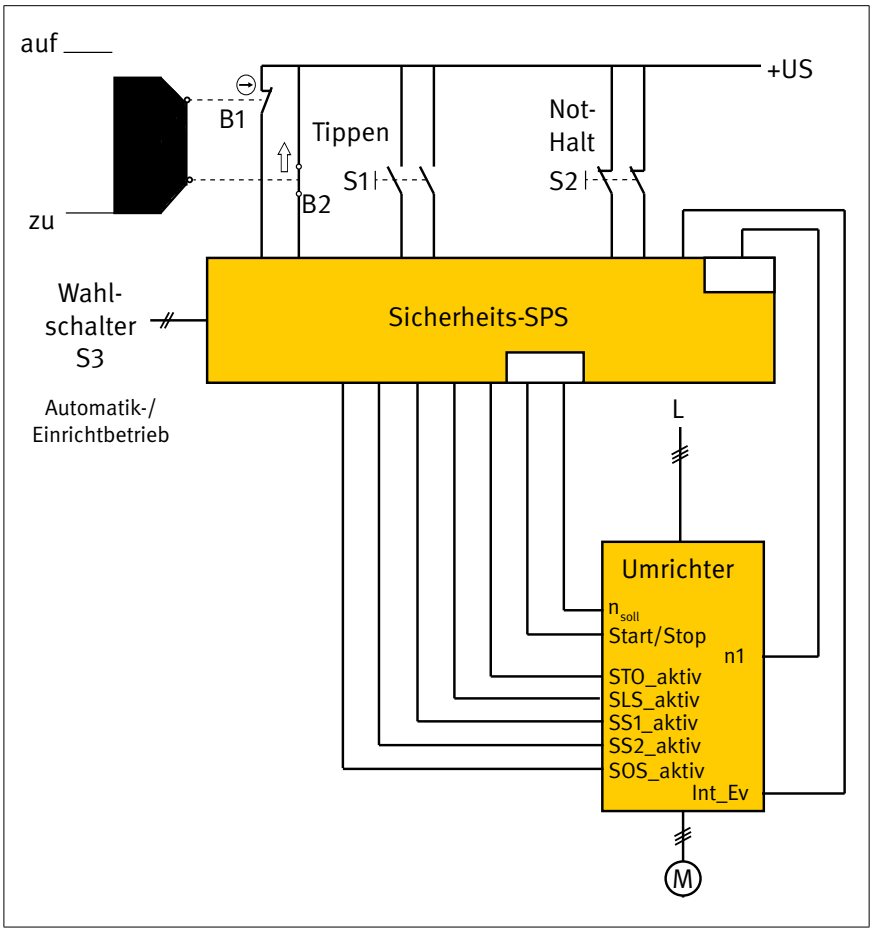



Abbildung 55:
Hardwareaufbau des Beispiels „SLS mit Sicherheits FU“

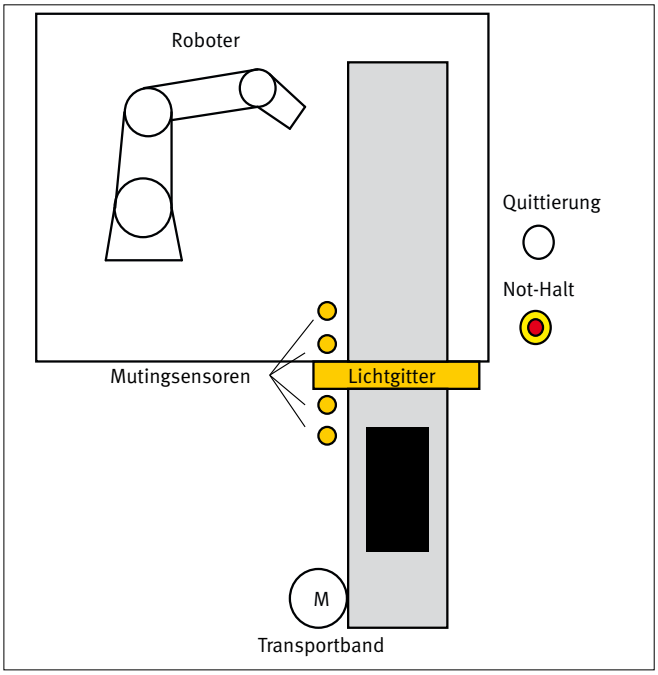
## 7.8 Muting

Das Beispiel „Muting“ behandelt eine Roboterfertigungszelle, in die das Material durch ein Lichtgitter zugeführt wird (Abbildung 56). Die Anlage hat nur die Betriebsart „Automatikbetrieb“.

Die Werkstücke werden von einem Transportband in den Fertigungsbereich des Roboters befördert. Dabei durchqueren sie ein Lichtgitter. Das Werkstück löst die Mutingsensoren in der richtigen Reihenfolge aus und kann deshalb das Lichtgitter durchfahren, ohne die Anlage zu stoppen. Nach der Bearbeitung fährt das Werkstück wieder aus dem Arbeitsbereich heraus.

Wird das Lichtgitter unterbrochen, ohne dass die Mutingsensoren korrekt ausgelöst werden, schaltet die Freigabe für das Transportband und den Roboter ab. Die Betätigung des Not-Halt-Tasters hat die gleiche Wirkung.

Abbildung 56:
Anlagenskizze einer Fertigungszelle mit Materialtransport durch ein Schutzgitter

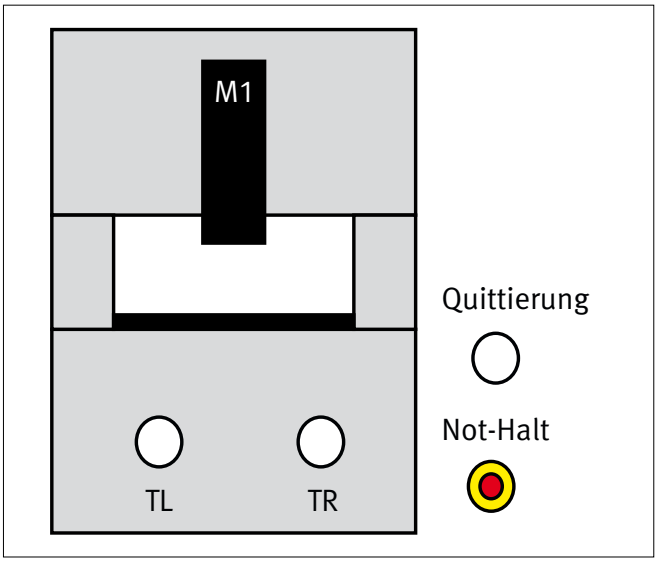



## 7.9 Zweihandbedienung

Das Beispiel „Zweihandbedienung“ behandelt eine Pressvorrichtung (Abbildung 57). Die Maschine hat nur die Betriebsart „Einrichtbetrieb“.

Die Bedienperson legt ein Werkstück in den Pressbereich und muss dann die beiden Taster der Zweihandbedienung gleichzeitig betätigen, um den Motor M1 freizugeben. Bei Betätigung des Not-Halt-Tasters wird der Motor sofort gestoppt.

Abbildung 57:
Anlagenskizze einer Pressvorrichtung



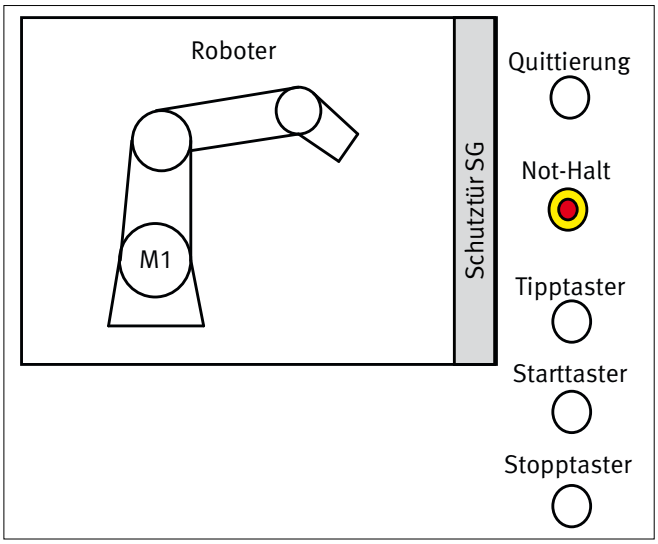
## 7.10 Konfigurierbares Schaltgerät

In diesem Beispiel überwacht ein konfigurierbares Schaltgerät eine Roboterfertigungszelle (Abbildung 58). Es gibt die Betriebsarten „Automatikbetrieb“ und „Einrichtbetrieb“.

Im Automatikbetrieb arbeitet der Roboter bei geschlossener Schutztür SG. Im Einrichtbetrieb darf der Roboter bei geöffneter Schutztür mit Zustimmung durch einen Tipptaster bei sicher begrenzter Geschwindigkeit (SLS) bewegt werden.

Dieses Beispiel soll veranschaulichen, dass die Matrixmethode des IFA zur Softwarespezifikation und Dokumentation von SRASW auch beim Einsatz von konfigurierbaren Schaltgeräten geeignet ist.

Abbildung 58:
Anlagenskizze einer Roboterfertigungszelle, überwacht mit konfigurierbarem Schaltgerät



# 8 Rolle der Embedded-Software für Anwendungsprogrammierung

Erst die in einer programmierbaren elektronischen Steuerung eingebettete Software (Embedded-Software, SRESW: Firmware, Laufzeitsystem, u. Ä.) ermöglicht die Entwicklung und Ausführung von SRASW. Dadurch trägt sie wesentlich zur Zuverlässigkeit der Sicherheitsfunktionen bei. DIN EN ISO 13849-1 unterscheidet bei ihren Anforderungen nicht zwischen programmierbaren Standard- oder Sicherheitskomponenten. Die Anforderungen an die Realisierung von SRESW und SRASW (Normenabschnitt 4.6) sind daher nicht nur für Sicherheitskomponenten, sondern auch für elektronische programmierbare Standardkomponenten (wie eine Standard-SPS) zu erfüllen.

Beim Einsatz einer zertifizierten Sicherheitssteuerung ist in diesem Kapitel nur Abschnitt 8.1 zu beachten!

## 8.1 Rolle der SRESW einer Sicherheitssteuerung

Bei einem Sicherheitsbauteil im Sinne der Maschinenrichtlinie ist die SRESW untrennbar mit der Steuerungselektronik verbunden und daher mit dieser zusammen meist nach DIN EN 61508 und DIN EN ISO 13849-1 entwickelt, geprüft und zertifiziert. Die Zuverlässigkeit dieser SRESW wird mit einem SIL oder dem Performance Level durch den Hersteller bescheinigt. Sicherheitsfunktionen mit $PL_r$ e dürfen durch programmierbare Steuerungen mit PL e realisiert werden und Entsprechendes gilt für niedrigere PL.

## 8.2 Bewertung der SRESW einer Standardsteuerung

Sicherheitsbezogene Steuerungen werden oft auch mit Standardkomponenten für den industriellen Anwendungsbereich realisiert. Im Vergleich zu geprüften und zertifizierten Sicherheitskomponenten ergeben sich jedoch Einschränkungen.
In jedem Fall ist auch die SRESW einer Standardkomponente zu bewerten. Entweder ist die SRESW selbst nach DIN EN ISO 13849-1, Abschnitt 4.6.2 entwickelt worden oder dies kann vom Hersteller der Standardkomponente bescheinigt werden. Beides ist eher selten der Fall; daher kann in der Regel nicht nachgewiesen werden, dass für eine programmierbare Standardkomponente die Anforderungen an die SRESW erfüllt werden. Wie geht man bei der Anwendungsprogrammierung damit um?

Während die Normversion DIN EN ISO 13849-1:2008 dazu noch keine konkreten Anforderungen nannte, findet sich in der Änderung 1 der DIN EN ISO 13849-1 [1], Abschnitt 4.6.2 folgende neue Anforderung:

*„Bauteile, für die die SRESW-Anforderungen nicht erfüllt sind, z. B. PLC(en) ohne Sicherheitsbewertung durch den Hersteller, dürfen unter folgenden alternativen Bedingungen verwendet werden:*

*- das SRP/CS ist auf PL a oder PL b begrenzt und verwendet Kategorie B, 2 oder 3;*

*- das SRP/CS ist auf PL c oder PL d begrenzt und darf mehrere Bauteile für zwei Kanäle in Kategorie 2 oder 3 verwenden. Die Bauteile dieser beiden Kanäle verwenden diversitäre Technologien.“*

**Achtung: Damit sind die Darstellungen im BGIA-Report 2/2008 [2], Abschnitt 6.3.10 veraltet.**

Diese neuen Anforderungen beziehen sich nur auf Embedded-Software. Bei der Verwendung von Bauteilen ohne Sicherheitsbewertung durch den Komponentenhersteller sind aber neben den Anforderungen an Embedded-Software nach DIN EN ISO 13849-1 viele weitere Anforderungen zu beachten, z. B.:

- hinsichtlich Vermeidung und Beherrschung systematischer Fehler oder Eignung für die zu erwartenden Umweltbedingungen, z. B. Klima, Vibration, elektromagnetischer Verträglichkeit.

- Für jedes in einem SRP/CS verwendete Bauteil sind mindestens die Anforderungen der Kategorie B einzuhalten (da alle Kategorien die Basisanforderungen von Kategorie B enthalten). Kategorie B erfordert u. a. die Übereinstimmung mit den zutreffenden Normen, also zum Beispiel mit EN 61131-2 für SPS, EN 61800-2 für Frequenzumrichter, Näherungsschalter 60947-5-2.

- Für jedes SRP/CS müssen systematische Ausfälle gemäß Anhang G berücksichtigt werden.

Die oben genannten neuen SRESW-Anforderungen gelten nicht nur für programmierbare Steuerungen ohne Sicherheitsbewertung durch den Hersteller, sondern für alle in einem SRP/CS verwendeten Bauteile, die Embedded-Software enthalten. Dies können z. B. auch Frequenzumrichter oder intelligente Sensoren (z. B. Geber, Näherungsschalter) sein.

Tabelle 23 zeigt die möglichen Kombinationen von PL und Kategorie mit Standardkomponenten und ob bzw. wie die Anforderungen an SRESW zu erfüllen sind.

Tabelle 23:
Anforderungen an die SRESW von Standardkomponenten (nach DIN EN ISO 13849-1:2015)

| Nr. | PL | Kategorie | Bedingungen | Anforderungen an SRESW der Standardkomponente |
|---|---|---|---|---|
| 1 | a oder b | B, 2, 3 | • Übereinstimmung mit zutreffenden Produktnormen<br>• qualitätsgesicherte Entwicklung als grundlegendes Sicherheitsprinzip | Keine SRESW-Anforderungen an industrielle Standardkomponenten gestellt |
| 2 | a, b, c | 1 | Generell keine Realisierung mit elektronischen Komponenten möglich, weil diese nicht als bewährte Bauteile nach DIN EN ISO 13849-1, Abs. 6.2.4 gelten | entfällt |
| 3 | c oder d | 2 oder 3 | • wie Nr. 1<br>• zwei Kanäle mit technologischer Diversität, SRASW realisiert erforderliche Fehlererkennung (DC) | Keine SRESW-Anforderungen an industrielle Standardkomponenten gestellt |
| 4 | c oder d | 2 oder 3 | Zwei Kanäle ohne technologische Diversität, SRASW realisiert erforderliche Fehlererkennung (DC) | Volle SRESW-Anforderungen nach DIN EN ISO 13849-1, Abschnitt 4.6.2 auch an industrielle Standardkomponenten gestellt. Sicherheitsbewertung durch den Komponentenhersteller erforderlich. |
| 5 | e | 3 oder 4 | PL e ist nach 13849-1, Abschnitt 4.6.2 für Standardkomponenten nicht möglich | entfällt |

Zu klären bleibt, was „Technologische Diversität“ bedeutet. Sie meint, dass aufgrund der Diversität (der technischen Unterschiedlichkeit) zweier Kanäle die Wahrscheinlichkeit eines gefährlichen Ausfalls des SRP/CS durch einen Fehler in der SRESW stark verringert wird. Hier sind systematische Ausfälle und Ausfälle infolge gemeinsamer Ursache relevant.

In den folgenden Beispielen kann „technologische Diversität“ üblicherweise als erfüllt angesehen werden:

- Ein Kanal (Funktionskanal oder Testkanal) enthält Bauteile mit Embedded-Software. Der zweite Kanal enthält ausschließlich Bauteile ohne Embedded-Software, also mechanische, elektronische, elektromechanische, pneumatische oder hydraulische Bauteile.

- Beide Kanäle benutzen diversitäre Embedded-Software, z. B. verschiedene Betriebssysteme, auf gleicher oder unterschiedlicher Hardware.
  Anmerkung: Bei Verwendung gleicher Hardware muss besonders auf die systematische Eignung der Bauteile für den geforderten Performance Level geachtet werden.

- Beide Kanäle verwenden unterschiedliche Hardware (z. B. Mikroprozessoren mit unterschiedlichen Prozessorkernen). Es wird angenommen, dass die Programmierung der zugehörigen eingebetteten Software in einer anderen Entwicklungsumgebung stattgefunden hat.

In den folgenden Beispielen kann „technologische Diversität“ üblicherweise nicht als erfüllt angesehen werden:

- Beide Kanäle benutzen gleichartige Bauteile von unterschiedlichen Herstellern ohne nähere Informationen zur Diversität der Embedded-Software. Hier kann üblicherweise nicht ausgeschlossen werden, dass beide Hersteller gleiche Embedded-Softwareteile benutzen, unter Umständen sogar auf identischer Hardware (Brandlabeling).

- Beide Kanäle verwenden Bauteile eines Herstellers unterschiedlichen Typs, ohne nähere Informationen zur Embedded-Software.

# 9 Einsatz von Standardsteuerungen für SRASW

Eine häufige Frage ist: Dürfen sicherheitsgerichtete Steuerungen mit Standardkomponenten wie z. B. einer Standard-SPS realisiert werden? Hier veröffentlichte das IFA 2011 ein Positionspapier [20] mit folgender Aussage: „*Sicherheitsrelevante Steuerungen können grundsätzlich durch den Einsatz von Standardkomponenten realisiert werden, jedoch bieten Sicherheitsbauteile den Vorteil, dass der Maschinenkonstrukteur bei der sicherheitstechnischen Beurteilung und Analyse der verwendeten Bauteile durch den Hersteller von Sicherheitsbauteilen entlastet wird. Zum Erreichen funktionaler Sicherheit ist neben der Verwendung einer geeigneten Architektur (Kategorie), der Realisierung einer erforderlichen Fehlererkennung und der Berücksichtigung von Ausfallraten/-wahrscheinlichkeiten die systematische Eignung von Komponenten zu beachten. Auszuschließen ist im Allgemeinen der Einsatz komplexer Elemente oder Teilsysteme gleichartiger Ausführung (homogene Redundanz), da Fragen nach der systematischen Eignung und der erforderlichen Fehleraufdeckung oft nicht ausreichend beantwortet werden können.*“

Standardkomponenten sind also grundsätzlich einsetzbar, aber mit erhöhtem Aufwand und mehr Verantwortung für denjenigen, der die Steuerung realisiert. Homogene Redundanz, z. B. durch Einsatz von zwei gleichen Standard-SPSen, ist wegen der kaum einschätzbaren Wahrscheinlichkeit systematischer Ausfälle von Hardware und SRESW (weil nicht geprüft oder zertifiziert) nicht geeignet. Details zu den Anforderungen bezüglich der SRESW der Standardkomponenten finden sich im Abschnitt 8.2.

In diesem Kapitel wird dargestellt, unter welchen Bedingungen die SRASW bei der Nutzung von Standardkomponenten zu entwickeln ist. Dabei wird vorausgesetzt, dass die programmierten Standardkomponenten – in Bezug auf Hardware und SRESW – auch eingesetzt werden dürfen.

## 9.1 Bestimmung der erforderlichen fehlervermeidenden Maßnahmen

Die Auswahl der fehlervermeidenden Maßnahmen für die SRASW in einer Standardkomponente orientiert sich wie auch bei Sicherheitskomponenten am erforderlichen Performance Level der realisierten Sicherheitsfunktion. Da aber bei zweikanaligen Architekturen durch Diversität in der Programmierung bzw. in der Technologie die kritischen Auswirkungen eines Softwarefehlers reduziert werden – und damit auch die Wahrscheinlichkeit eines gefährlichen Ausfall der Sicherheitsfunktion abnimmt – können als Empfehlung des IFA die Anforderungen bzw. deren Wirksamkeit in einigen der nachfolgend dargestellten Fälle um eine $PL_r$-Stufe reduziert werden. Dies lässt sich aus Abschnitt 7.4.3 „Synthese von Elementen zum Erreichen der erforderlichen systematischen Eignung“ der DIN EN 61508-2:2010 ableiten.

## 9.2 Einkanalige Architekturen

Einkanalige Architekturen sind der klassische Anwendungsfall für eine Standardkomponente wie z. B. Standard-SPS: eine einkanalige Architektur einer Kategorie B für einen maximal möglichen $PL_r$ b. Für die Komponente muss kein Diagnosedeckungsgrad erreicht werden. Die SRASW ist unter Anwendung der fehlervermeidenden Basismaßnahmen (Abschnitt 3.2) zu realisieren. Die Matrixmethode des IFA kann hier eingesetzt werden (Abschnitt 9.4).

Kategorie 1 ist ebenfalls einkanalig, erfordert aber nach DIN EN ISO 13849-1, Abschnitt 6.2.4 bewährte Bauteile. Damit sind komplexe elektronische Komponenten ausgeschlossen. $PL_r$ c kann also nur mit zweikanaligen Architekturen realisiert werden.

## 9.3 Zweikanalige Architekturen

Wenn bei zweikanaligen Architekturen je Kanal eine programmierbare Standardkomponente eingesetzt wird, kann die SRASW der beiden Komponenten entweder gleich oder diversitär sein. Diese beiden Fälle werden im Folgenden unterschieden. In diesem Zusammenhang zählt auch Kategorie 2 zu den zweikanaligen Architekturen (ein Funktionskanal und ein Testkanal).

### 9.3.1 Merkmale diversitärer SRASW

Diversitäre SRASW bedeutet, dass zwei (oder mehr) Programme auf der Basis derselben Spezifikation die gleichen Aufgaben erfüllen sollen, aber unterschiedlich entwickelt wurden. Die Wahrscheinlichkeit eines gefährlichen Ausfalls durch gleichzeitiges Auftreten eines systematischen Fehlers in den SRASW soll dadurch reduziert werden. Merkmale diversitärer SRASW können z. B. sein:

- unterschiedliche Programmierende bzw. Programmierteams,
- unterschiedliche Entwürfe (Softwarestruktur, Algorithmen etc.) durch dieselben Personen realisiert,
- unterschiedliche Programmiersprachen: textuelle Sprache (ST) vs. grafische Sprache (FBD),
- unterschiedliche Entwicklungsumgebungen: freie Programmierung einer SPS vs. grafische Konfiguration eines Steuerrelais.

### 9.3.2 Beide Kanäle mit gleicher, homogener SRASW

Wird eine programmierte Standardkomponente in einem Kanal des Steuerungsteils in Redundanz mit einer anderen programmierten Standardkomponente in dem anderen Kanal eingesetzt und beide SRASW sind nicht diversitär programmiert (Merkmale siehe Abschnitt 9.3.1), dann gelten aufgrund der

Wahrscheinlichkeit eines gleichzeitigen gefährlichen Ausfalls durch systematische Fehler für beide SRASW die vollen normativen Anforderungen des erforderlichen PL (Abschnitt 3.2). Unabhängig davon sind auch die normativen Anforderungen an die SRESW einzuhalten (Abschnitt 8.2).

#### 9.3.3 Beide Kanäle mit diversitärer SRASW

Wird eine programmierte Standardkomponente in einem Kanal des Steuerungsteils in Redundanz mit einer anderen programmierten Standardkomponente in dem anderen Kanal in Kategorie 3 oder 4 eingesetzt und sind die beiden SRASW diversitär programmiert (siehe Abschnitt 9.3.1), dann können aufgrund der geringeren Wahrscheinlichkeit eines gleichzeitigen gefährlichen Ausfalls durch systematische Fehler in den beiden SRASW die normativen Anforderungen für beide SRASW um eine PL-Stufe abgesenkt werden (z. B. anstelle von Anforderungen für $PL_r$ d dann für $PL_r$ c, siehe Abschnitt 3.2). Bei Kategorie 2 können nur die Anforderungen für die SRASW des Testkanals abgesenkt werden. Unabhängig davon sind auch die normativen Anforderungen an die SRESW einzuhalten (Abschnitt 8.2).

#### 9.3.4 Nur ein Kanal mit SRASW

Wird eine programmierte Standardkomponente in einem Kanal des Steuerungsteils in diversitärer Redundanz mit einer anderen Technologie als der programmierbar-elektronischen (z. B. fluidtechnisch) in dem anderen Kanal in Kategorie 3 oder 4 eingesetzt, dann können aufgrund der geringeren Wahrscheinlichkeit eines gefährlichen Ausfalls durch systematische Fehler in dieser SRASW die normativen Anforderungen um eine PL-Stufe abgesenkt werden (z. B. anstelle von Anforderungen für $PL_r$ d dann für $PL_r$ c, siehe Abschnitt 3.2). Bei Kategorie 2 können nur die Anforderungen für die SRASW des Testkanals abgesenkt werden. Unabhängig davon sind auch die normativen Anforderungen an die SRESW einzuhalten (Abschnitt 8.2).

## 9.4 Anwendung der Matrixmethode des IFA auf Standardkomponenten

Die hier vorgestellte Matrixmethode des IFA kann auch auf die Anwendungsprogrammierung von Standardkomponenten angewendet werden, solange die Softwarestruktur wie in Abbildung 10 den drei Stufen „Vorverarbeitungsebene“ ⇢ „Ansteuerlogik“ ⇢ „Ansteuerungsebene“ entspricht und die fehlererkennenden Maßnahmen in der Vorverarbeitungs- und in der Ansteuerungsebene implementiert sind. Dies gilt umso mehr bei der Verwendung von zwei Standardkomponenten in zwei Kanälen mit diversitärer Anwendungsprogrammierung. Ein gutes Beispiel dafür liefert das Validierungsbeispiel in DIN EN ISO 13849-2 [10], Anhang E. Hierfür sind auch Beispielprojekte verfügbar (siehe Abschnitt 12.3).

## 9.5 Einsatz von Standardkomponenten für fehlerbeherrschende Maßnahmen

Häufig werden Standardkomponenten auch zur Realisierung von Testeinrichtungen eingesetzt. Es stellt sich die Frage nach der notwendigen Zuverlässigkeit dieser Standardkomponenten – einkanalig oder zweikanalig? Grundsätzlich sollte gelten, dass die Testeinrichtung nicht wesentlich früher als die von ihr überwachten Komponenten ausfällt. Andererseits ist es aber auch nicht effektiv, viel mehr in die Zuverlässigkeit der Testeinrichtung zu investieren als in die Steuerungsteile, die die eigentliche Sicherheitsfunktion ausführen.

DIN EN ISO 13849-1 hält sich daher mit Anforderungen an die Zuverlässigkeit der Testeinrichtungen zurück. Bei den Kategorien 3 und 4 wird auf die Einfehlertoleranz vertraut, da inklusive des Ausfalls der Testeinrichtung insgesamt mehrere gefahrbringende Ausfälle notwendig sind, bevor die Sicherheitsfunktion nicht mehr ausgeführt wird. Dass dieser Fall unbemerkt auftreten kann, wird als extrem unwahrscheinlich angesehen. Daher können Testeinrichtungen einkanalig – mit Standardkomponenten, wie z. B. eine Standard-SPS – ohne spezielle Anforderungen an deren Zuverlässigkeit realisiert werden. Beispiele finden sich im BGIA-Report 2/2008 [2]: Beispiele 22 (Bauteil K4), 28 (K3) und 33 (K1).

Bei Kategorie 2 gibt es zumindest bei der vereinfachten PL-Bestimmung (DIN EN ISO 13849-1, Abschnitt 4.5.4) eine Nebenbedingung, die bei der $PFH_D$-Berechnung zugrunde gelegt wurde: Hier darf die gefahrbringende Ausfallrate der Testeinrichtung (z. B. der Standard-SPS) nicht mehr als doppelt so hoch sein wie die gefahrbringende Ausfallrate der davon überwachten Komponenten. Im Zweifel lässt sich dieser Vergleich kanalweise durchführen, sodass der $MTTF_D$-Wert des gesamten Testkanals nicht kleiner sein sollte als der halbe $MTTF_D$-Wert des Funktionskanals. Auch dafür finden sich Beispiele im BGIA-Report 2/2008: Beispiele 9 (Bauteil K3), 11 (K1) und 12 (K1). Bezüglich der SRASW sind die normativen Anforderungen zu beachten und die Fallbeispiele in Abschnitt 9.3 zu berücksichtigen: Bei diversitärer SRASW (bzw. SRASW nur im Testkanal) können die Anforderungen um eine $PL_r$-Stufe abgesenkt werden.

Kategorie B und 1 erfordern keine Testung.

# 10 Typische Test- und Überwachungsmaßnahmen in SRASW

Bei zertifizierten Sicherheitssteuerungen und den mitgelieferten Hersteller-Funktionsbausteinen werden fehlerbeherrschende Selbsttest- und Online-Überwachungsmaßnahmen schon angemessen vom Steuerungshersteller implementiert. Dieses Kapitel gibt dagegen Hinweise in Bezug auf Standardkomponenten bzw. Anwender-Funktionsbausteine, bei denen fehlerbeherrschende Maßnahmen erst im Rahmen der Anwendungsprogrammierung realisiert werden können.

Ein wichtiger Teil der SRASW sind die Test- und Überwachungsmaßnahmen für zufällige und systematische Steuerungsausfälle. Durch wirksame Tests lässt sich z. B. eine schlechte Zuverlässigkeit der (Peripherie-)Komponenten teilweise kompensieren und damit der Performance Level der Sicherheitsfunktion erhöhen. Die Güte der Fehlererkennung wird in DIN EN ISO 13849-1 mit dem Diagnosedeckungsgrad DC (Diagnostic Coverage) angegeben. In Anhang E der Norm findet sich eine ausführliche Liste typischer Testmaßnahmen und des jeweils erreichbaren DC.

Im Kontrast zu den fehlervermeidenden Maßnahmen (Kapitel 5) werden diese Test- und Überwachungsmaßnahmen auch als fehlerbeherrschende Maßnahmen bezeichnet. Oft sind Selbsttests schon in der SRESW (Embedded-Software) realisiert, um die Ausfälle der programmierbaren Steuerung selbst zu beherrschen. In der SRASW sind darüber hinaus auch Tests der angeschlossenen Peripheriegeräte, z. B. der Sensoren oder der Aktoren, zu programmieren. Typischerweise sind diese Tests bereits in den Hersteller-Funktionsbausteinen der Vorverarbeitungs- und Ansteuerungsebene implementiert (Abschnitt 6.17). Darüber hinaus müssen projekt- bzw. maschinenspezifische Funktionsbausteine mit Tests und Überwachungen auch selbst in der Anwendung programmiert werden. Dieses Kapitel soll dazu Informationen liefern.

## 10.1 Typische Techniken für Tests und Überwachungen

Zum Testen der angeschlossenen Peripheriegeräte gibt es einige typische Techniken, bei denen immer eine Erwartungshaltung (z. B. aufgrund eines Signalwechsels) existiert, gegen die ein anderes Signal überprüft wird:

- Kreuzvergleich zweier Logikkomponenten: Zwei Logikkomponenten eines Steuerungsteils (z. B. zwei Standard-SPSen) tauschen ihre Eingangssignale und Verarbeitungsergebnisse aus und vergleichen diese gegenseitig.

- Logische und zeitliche Programmlaufüberwachung: Eine Testeinrichtung kontrolliert, ob alle wichtigen Programmmodule der überwachten Steuerung in der erwarteten Zeit- und Reihenfolge ausgeführt werden.

- Plausibilitätsprüfung von Sensorkomponenten: Ein Sensorsignal (binär oder analog) wird mit einem zweiten Sensorsignal von derselben Messstelle verglichen. Ein analoges Sensorsignal wird mit einem erwarteten Signalbereich verglichen.

- Direkte Überwachung: Eine Aktorkomponente (Schütz/Ventil) wird über ein Ausgangssignal der Logik geschaltet. Direkt an der Aktorkomponente wird die Schalthandlung durch eine Diagnosekomponente, z. B. elektrischer Kontakt, registriert und an die Logik zurückgemeldet. Nach Abwarten der Diskrepanzzeit wird das Schaltsignal mit dem Rückmeldesignal verglichen, um das erfolgreiche Schalten zu überwachen.

- Indirekte Überwachung: Eine Aktorkomponente (Umrichter/Schütz/Ventil) wird über ein Ausgangssignal der Logik geschaltet. An der Aktorkomponente kann die Schalthandlung nicht direkt gemessen werden. Stattdessen wird nur die Auswirkung des Schaltens durch eine Diagnosekomponente an nachgeordneten Steuerungskomponenten, Verbindungsmitteln oder Aktoren (z. B. Druckschalter/Drehgeber/ Endschalter/ Wegaufnehmer) registriert und an die Logik zurückgemeldet. Nach Abwarten der Diskrepanzzeit wird das Schaltsignal mit dem Rückmeldesignal verglichen, um das erfolgreiche Schalten zu überwachen.

Aufgrund zeitlicher Verzögerungen der Signale muss immer eine Toleranzzeit, die sogenannte Diskrepanzzeit, abgewartet werden, bevor das Vergleichsergebnis gültig ist. Bei analogen Signalen muss eine technisch bedingte Wertetoleranz berücksichtigt werden.

## 10.2 Randbedingungen zu Test- und Überwachungsmaßnahmen

Die Erkennung eines gefahrbringenden Ausfalls ist nur der Anfang eines erfolgreichen Tests. Zusätzlich ist die Einleitung eines sicheren Zustands, aus dem heraus keine Gefährdung mehr besteht, erforderlich. Dazu gehört ein wirksamer Abschaltpfad, was z. B. bei einkanalig getesteten Systemen (Kategorie 2) dazu führt, dass ein zweites Abschaltelement vorhanden sein muss. Dieses ist nötig, um den sicheren Zustand einzuleiten oder aufrechtzuerhalten, wenn der Test ein Versagen des regulären Abschaltelements festgestellt hat. Bei zweikanaligen Systemen (Kategorie 3 und 4) wird bei Versagen eines Kanals der sichere Zustand durch den zweiten Abschaltpfad eingeleitet.

Sowohl das Auslösen eines Tests, dessen Ausführung sowie die erforderliche Abschaltung sollten bevorzugt automatisch von der SRASW durchgeführt werden. Nur in Ausnahmefällen erscheint es angeraten, hier auf eine manuelle Intervention, z. B. des Maschinenbedienpersonals, angewiesen zu sein. Gleichwohl berücksichtigt die Bestimmung des

Diagnosedeckungsgrads für zweikanalige Systeme eine Fehlererkennung bei Anforderung der Sicherheitsfunktion, das heißt, es werden nicht nur automatisch ausgelöste Tests in der SRASW betrachtet. Gerade bei elektromechanischen Bauteilen, z. B. Relais oder Schützen, kann eine Erkennung des Fehlers üblicherweise nur bei Anforderung der Sicherheitsfunktion erfolgen. Für den Diagnosedeckungsgrad der Fehlererkennung bei Anforderung muss die Häufigkeit der Anforderung der Sicherheitsfunktion berücksichtigt werden.

## 10.3 Testhäufigkeit

Ein weiterer Aspekt ist die Frage nach der notwendigen Testhäufigkeit. Ein zu selten ausgeführter Test wird unter Umständen durch das Eintreten eines Gefährdungsereignisses überholt und bietet damit nur trügerische Sicherheit. Als Faustregel gilt: Die Testhäufigkeit konkurriert immer mit anderen Häufigkeiten, daher kann eine ausreichende Häufigkeit nicht generell genannt werden. In zweikanaligen Systemen der Kategorien 3 und 4 steht die Testhäufigkeit in Konkurrenz zur Häufigkeit des Auftretens eines zweiten gefahrbringenden Ausfalls. Denn erst wenn der zweite Kanal ausfällt, bevor ein Test den Ausfall des ersten bemerkt hat, besteht die Gefahr der Nichtausführung der Sicherheitsfunktion. Systeme der Kategorie 4 tolerieren gemäß Definition sogar die Anhäufung unerkannter Fehler. In zweikanaligen Systemen hat sich ein Test einmal pro Schicht in der Praxis bewährt.

Anders ist es beim einkanalig getesteten System der Kategorie 2: Hier muss der Test erfolgreich sein, bevor die nächste Anforderung der Sicherheitsfunktion – also eine potenzielle Gefährdung – erfolgt. Hier steht die Testhäufigkeit also in Konkurrenz zur Häufigkeit der Anforderung der Sicherheitsfunktion. Ein Faktor von 100 wird als ausreichend angesehen, also eine mindestens 100-mal höhere Testrate als die mittlere Anforderungsrate der Sicherheitsfunktion. Wo ein Faktor von 100 nicht möglich ist, kann die Testrate bis zu einem Faktor von 25 reduziert werden. Dann ergibt sich gegenüber dem Faktor 100 eine Erhöhung der Ausfallwahrscheinlichkeit $PFH_D$ von ca. 10 % [21].

Falls in einkanalig getesteten Systemen allerdings die Tests so schnell ausgeführt werden, dass der sichere Zustand erreicht wird, bevor es zu einer Gefährdung kommt, werden keine Bedingungen an die Testhäufigkeit gestellt (DIN EN ISO 13849-1:2016, Abschnitt 4.5.4, Anforderungen an Kategorie 2).

## 10.4 Weiterführende Informationen

Anregungen für Testmaßnahmen finden sich beispielsweise in der Literatur [8; 22 bis 24].

# 11 Kombinationen mehrerer Steuerungsteile mit Software

Bisher war in diesem Report nur die Rede von einem programmierbaren Steuerungsteil (SRP/CS) mit einem Anwendungsprogramm als Teil einer Sicherheitsfunktion. Manchmal ist es aber notwendig, mehrere Steuerungsteile mit ihren jeweiligen Anwendungsprogrammen als Subsysteme hintereinander zu schalten, die jeweils in Teilen die Sicherheitsfunktion ausführen (z. B. SPS und programmierbarer Antriebsregler). Bei der Bewertung der Steuerungshardware, also der Wahrscheinlichkeit eines gefährlichen, zufälligen Bauteileausfalles, werden die Ausfallwahrscheinlichkeiten ($PFH_D$) der einzelnen Steuerungsteile addiert, sodass typischerweise mit jedem zusätzlichen Steuerungsteil die Zuverlässigkeit der Hardware abnimmt.

Was ist nun in Bezug auf SRASW und den systematischen Ausfällen zu beachten, wenn mehrere programmierte Steuerungsteile für eine Sicherheitsfunktion kombiniert werden? Hat das Einfluss auf die Anforderungen oder den Entwicklungsablauf?

Die Kombination mehrerer Steuerungsteile behandelt die DIN EN ISO 13849-1 [1] im Abschnitt 6.3 mit dem Ziel, für die Kombination und damit für die realisierte Sicherheitsfunktion einen Gesamt-PL zu bestimmen. Gleichzeitig wird eine Gesamt-$PFH_D$ ermittelt, indem die einzelnen $PFH_D$-Werte der Steuerungsteile zu addieren sind. Explizite Anforderungen oder Vorgaben an die Kombination von Anwendungsprogrammen dieser Steuerungsteile finden sich in dem Abschnitt jedoch nicht. Die Norm stellt auch fest: „*Der PL der kombinierten SRP/CS wird beschränkt durch den niedrigsten PL eines einzelnen SRP/CSi* [Steuerungsteils]*, das an der Durchführung der Sicherheitsfunktion beteiligt ist, da der PL auch durch nicht quantifizierbare Aspekte* [wie z. B. Softwarequalität] *bestimmt wird; …*“

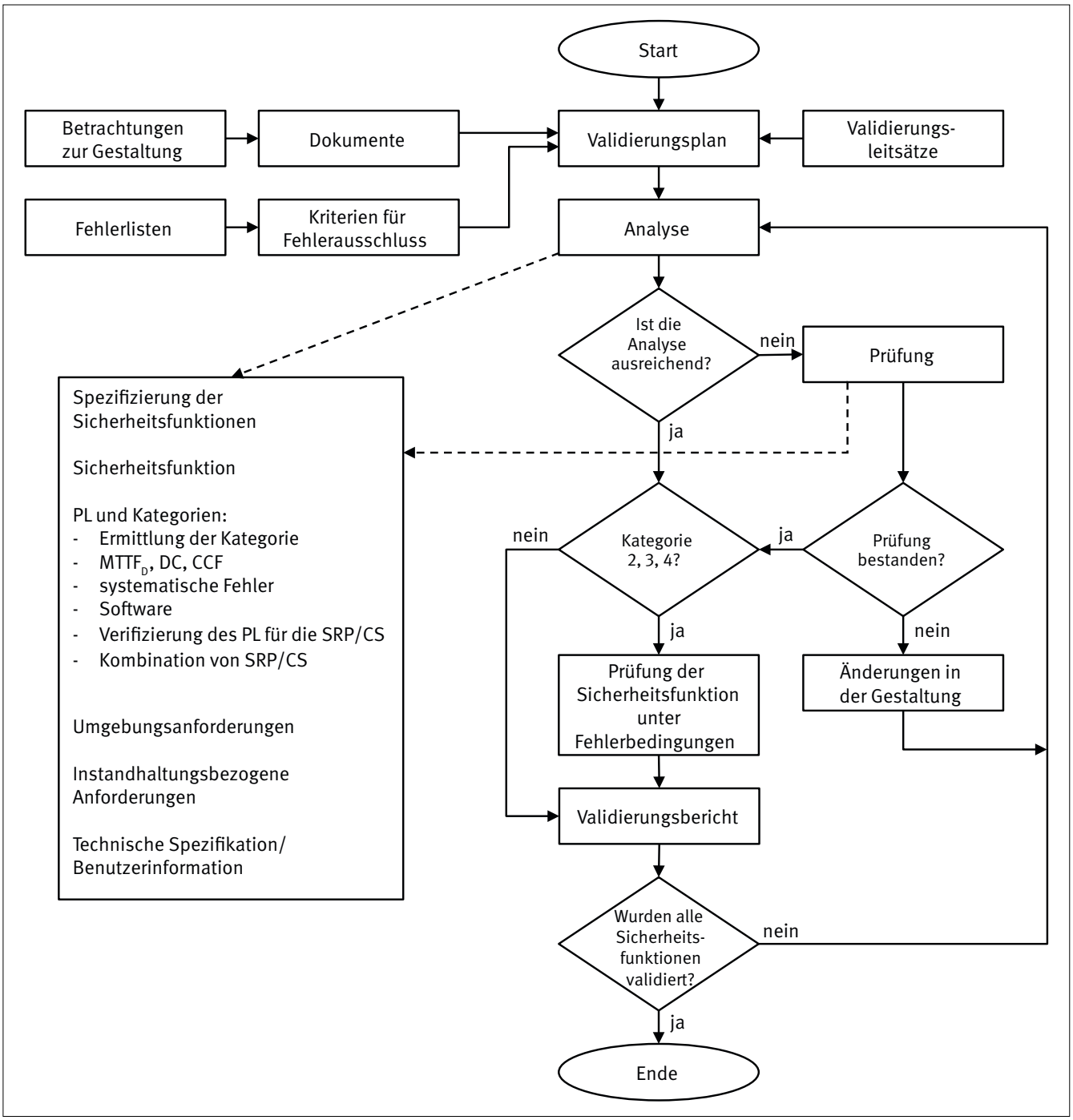
Damit bestimmt die „schlechteste“ Anwendungssoftware den erreichbaren PL für die gesamte Kombination bzw. Sicherheitsfunktion, selbst wenn die Gesamt-$PFH_D$ sehr klein sein sollte. Dies ist auch in Abschnitt 3.2 mit Abbildung 3 dargestellt.

Umgekehrt bedeutet dies auch, dass durch die Vorgabe eines $PL_r$ für eine Sicherheitsfunktion dann für alle Anwendungsprogramme der Steuerungsteile dieselben Anforderungen nach Abschnitt 4.6 der Norm gelten. Es ist natürlich zu prüfen, ob durch die Reihenschaltung der Steuerungsteile mit ihren Anwendungsprogrammen aufgrund von eventuellen negativen Wechselwirkungen der Programme untereinander sich die Wahrscheinlichkeit für systematische Ausfälle durch die Programme erhöhen könnte.

# 12 Validierung von SRASW

Validierung bezeichnet qualitätssichernde Maßnahmen zur Vermeidung von Fehlern während des Entwurfes und der Realisierung sicherheitsbezogener Teile von Steuerungen (SRP/CS), die Sicherheitsfunktionen ausführen. Besonders ausgiebig beschäftigt sich DIN EN ISO 13849-2:2013 [10] mit diesem Thema. Validierung kann durch alleinige Analyse oder durch eine Kombination aus Analyse und Tests[1] erfolgen. Abbildung 59 gibt einen Überblick über das Validierungsverfahren.

Abbildung 59:
Übersicht über das Validierungsverfahren (nach DIN EN ISO 13849-2:2013)




[1] In der deutschen Übersetzung der EN ISO 13849-2:2013 wurde der englische Begriff „testing“ durch Prüfung übersetzt. In anderen Normen der funktionalen Sicherheit ist dagegen die Übersetzung „Test“ gebräuchlich und wird auch in diesem Report verwendet. Die Aktivität „Prüfen“ wird dagegen von internen bzw. externen Prüfstellen wahrgenommen.

Die Validierung ist der Nachweis der Eignung, bezogen auf den realen Einsatzzweck, der während oder am Ende des Entwicklungsprozesses erfolgt. Es wird überprüft, ob die spezifizierten Sicherheitsanforderungen an den sicherheitsrelevanten Teilen der Maschinensteuerung erreicht wurden. Auf SRASW bezogen wäre dies z. B. die Überprüfung, ob die Sicherheitsfunktionen a) wie spezifiziert und b) in der durch den $PL_r$ geforderten Qualität durch die SRASW realisiert wurden. Für die Kategorien (mit Fehlererkennung) 2, 3 und 4 muss die Validierung der Sicherheitsfunktion auch Tests unter Fehlerbedingungen umfassen (Erweiterter Funktionstest, Abschnitt 5.10). So werden auch fehlererkennende und -beherrschende Teile der SRASW mitgetestet.

## 12.1 Allgemeine Anforderungen zur Validierung

In DIN EN ISO 13849-2:2013 sind zunächst allgemeine Anforderungen für alle Aspekte der Validierung formuliert – einige davon mit Bezug zu SRASW. Diese allgemeinen Anforderungen werden im Folgenden kurz dargestellt. Die Validierung sollten Personen durchführen, die unabhängig von der Gestaltung der Steuerungsteile sind (siehe auch Abschnitt 5.15).

Bei Anwendung der Matrixmethode des IFA mit dem Tool SOFTEMA (Kapitel 14) werden diese allgemeinen Anforderungen grundsätzlich umgesetzt. Zusätzliche Informationen, z. B. über Umgebungsbedingungen oder Testausrüstung können als Dokumente mit SOFTEMA verknüpft werden.

### 12.1.1 Validierung durch Analyse und Tests

Die Validierung einer SRASW besteht aus der Analyse (siehe DIN EN ISO 13849-2, Abschnitt 5) und aus der Durchführung von Funktionstests (siehe DIN EN ISO 13849-2, Abschnitt 6) unter vorhersehbaren Bedingungen in Übereinstimmung mit dem Validierungsplan.

DIN EN ISO 13849-2 beschreibt in Kapitel 5 die allgemeine Vorgehensweise sowie Techniken der Analyse. Es heißt dort: „*Die Validierung von SRP/CS muss durch eine Analyse erfolgen*“, die u. a. auch die nicht quantifizierbaren, qualitativen (Software-)Aspekte – die das Systemverhalten beeinträchtigen – betrachten.

Bei der Analyse soll anhand der Durchsicht von Unterlagen, z. B. mit Review oder Walk through, und ggf. durch den Einsatz von Analysewerkzeugen, z. B. Tools zur statischen und dynamischen Softwareanalyse oder FMEA-Tools, festgestellt werden, ob die spezifizierten Anforderungen erreicht wurden.

Mit der Analyse sollte so früh wie möglich und gleichzeitig mit dem Entwicklungsprozess begonnen werden, sodass Probleme frühzeitig korrigiert werden können. Für einige Teile der Analyse kann es notwendig sein, sie erst dann auszuführen, wenn die Entwicklung weit fortgeschritten ist. Konkrete Analyseschritte bezogen auf Software sind in diesem Report im Abschnitt 12.2 dargestellt.

DIN EN ISO 13849-2 beschreibt dann in Kapitel 6 die allgemeine Vorgehensweise sowie Techniken der Prüfung. Zur Notwendigkeit der Prüfung heißt es: „*Wenn die Validierung durch Analyse nicht schlüssig ist, müssen Prüfungen* [Tests] *durchgeführt werden, um die Validierung zu vervollständigen. Eine Prüfung* [Test] *als Ergänzung zur Analyse ist oft notwendig.*“

Zur Anzahl der Testobjekte heißt es: „*Soweit nicht anders festgelegt, müssen die Prüfungen* [Tests] *an einem einzelnen Produktionsmuster des zu prüfenden* [testenden] *sicherheitsbezogenen Teils durchgeführt werden.*“ Für SRASW genügt also ein Produktionsmuster zum Test.

Es folgt eine Zusammenfassung des Kapitels 6 mit den Anforderungen an die Validierungsprüfungen:

- Tests müssen geplant, in logischer Reihenfolge ausgeführt und aufgezeichnet werden.
- Der Testplan enthält die Testspezifikationen, die erwarteten Ergebnisse und die zeitliche Abfolge der Tests.
- Die Testaufzeichnungen enthalten die Namen der testenden Personen, die Umgebungsbedingungen, den Testablauf und die verwendete Ausrüstung, das Testdatum und die Ergebnisse des Tests.
- Die Testaufzeichnungen müssen mit dem Testplan verglichen werden, um sicherzustellen, dass die festgelegten Funktions- und Leistungsziele erreicht sind.
- Der Test am Testobjekt muss so nah wie möglich in der vorgesehenen endgültigen Betriebskonfiguration, d. h. mit allen peripheren Geräten und angebrachten Abdeckungen, durchgeführt werden.
- Die Tests dürfen manuell oder automatisch (z. B. durch Computer) durchgeführt werden.
- Sofern angebracht, muss die Validierung der Sicherheitsfunktionen durch Tests durchgeführt werden, bei denen Eingangssignale in verschiedenen Kombinationen in die SRP/CS eingegeben werden. Die sich ergebende Reaktion an den Ausgängen muss mit den spezifizierten Ausgangssignalen verglichen werden.
- Es wird empfohlen, die Kombination dieser Eingangssignale systematisch in die Steuerung und Maschine einzugeben. Ein Beispiel für diese Logik ist: Energie einschalten, in Betrieb setzen, Arbeitsablauf, Richtungsänderungen, Wiederanlaufen. Falls notwendig, muss ein erweiterter Umfang von Eingangsdaten eingegeben werden, um anomale oder ungewöhnliche Situationen zu berücksichtigen und um zu sehen, wie die SRP/CS reagieren. Derartige Kombinationen von Eingangsdaten müssen vorhersehbare fehlerhafte Bedienungen berücksichtigen. Das Tool SOFTEMA unterstützt die Dokumentation dieser Testfälle.

### 12.1.2 Validierungsplan

Die Validierung muss gemäß DIN EN ISO 13849-2 geplant und dokumentiert werden. Dabei ist Folgendes darzustellen:

- die Identität der Dokumente für die Spezifikationen,
- die Betriebs- und Umgebungsbedingungen während der Tests,
- die anzuwendenden Analysen und Tests,
- den Verweis auf anzuwendende Test-/Prüfnormen und
- die für jeden Schritt im Validierungsprozess verantwortlichen Personen oder Parteien.

### 12.1.3 Angaben zur Validierung

Um SRASW validieren zu können, sind gemäß DIN EN ISO 13849-2 typischerweise folgende Dokumente notwendig:

- Spezifikation der erforderlichen Eigenschaften jeder Sicherheitsfunktion und ihrer/ihres erforderlichen Kategorie und Performance Levels,
- Blockdiagramm(e) mit einer Funktionsbeschreibung der Blöcke,
- Schaltpläne einschließlich ihrer Verknüpfungen/Verbindungen,
- Funktionsbeschreibung der Schaltpläne,
- Ablaufdiagramm(e) für schaltende Bauteile und Signale, die sicherheitsrelevant sind,
- falls relevant: Benutzerinformation, z. B. Anleitung für Installation und Betrieb/Benutzerhandbuch.

Die Software-Dokumentation muss Folgendes enthalten:

- eine Spezifikation, die klar und eindeutig ist, und die die sicherheitstechnische Leistungsfähigkeit, die die Software erreichen muss, angibt,
- den Nachweis, dass die Software so gestaltet ist, dass sie den erforderlichen Performance Level erreicht (siehe Abschnitt 12.2) und
- Einzelheiten über Tests (insbesondere Testberichte), die durchgeführt wurden, um nachzuweisen, dass die geforderte sicherheitstechnische Leistungsfähigkeit erreicht wurde.

### 12.1.4 Validierungsaufzeichnung

Die Validierung durch Analyse und Tests muss aufgezeichnet werden. Die Aufzeichnung bzw. das Protokoll muss das Validierungsverfahren für alle sicherheitstechnischen Anforderungen ersichtlich machen. Querverweise zu vorhergehenden, eindeutig gekennzeichneten Aufzeichnungen sind möglich. Sicherheitsbezogene Teile, die die Validierung nicht bestanden haben, müssen ebenfalls dokumentiert werden. Es ist sicherzustellen, dass sämtliche Teile nach einer Veränderung erfolgreich neu validiert werden.

## 12.2 Spezielle Anforderungen zur Validierung von SRASW

In DIN EN ISO 13849-2, Abschnitt 9.5, sind neben den allgemeinen Anforderungen auch spezielle Anforderungen zur Validierung von SRASW formuliert. Diese sollen hier vorgestellt werden, auch in Hinblick darauf, wie diese in der Matrixmethode des IFA mit dem Tool SOFTEMA (Kapitel 14) umgesetzt werden können. Zunächst ist der Umfang der Validierung in Tabelle 24 dargestellt.

Tabelle 24:
Umfang der Validierung von SRASW nach DIN EN ISO 13849-2:2013

| Anforderung nach DIN EN ISO 13849-2 | Umsetzungsschritte in der Matrixmethode des IFA (Dokumente) | Reportabschnitte |
|---|---|---|
| festgelegte Funktionsverhalten und Leistungskriterien, wenn auf der Zielhardware ausgeführt | 1) Verifikation C&E-Matrix (B4) gegen Spezifikation Sicherheitsfunktionen (A1) | 6.3, 6.7 |
| | 2) Verifikation Code gegen C&E-Matrix (B4) | 6.7 |
| | 3) Funktionsprüfung SRASW auf Zielhardware gegen C&E-Matrix (B4) | 6.7 |
| | 4) Protokoll Codereview (C1) und Validierung (D1) | 6.8 |
| ausreichende Softwaremaßnahmen für den festgelegten $PL_r$ der Sicherheitsfunktion(en) | 1) Auflistung Sicherheitsfunktionen mit $PL_r$ und Feststellung des maximalen $PL_r$ (A1) | 6.3 |
| | 2) Auswahl der Softwaremaßnahmen anhand $PL_r$ im Katalog Maßnahmen (A3) | 6.5 |
| | 3) Verifikation der Umsetzung der Maßnahmen während der Codierung (auch in A3) | 6.5 |
| | 4) Protokoll Codereview (C1) | 6.8 |
| angewandte Maßnahmen und Methoden zur Vermeidung von systematischen Softwarefehlern während der Softwareentwicklung | 1) Auswahl und Anwendung von Maßnahmen (A3) und normativer Anforderungen (A4) | 6.5 |
| | 2) Durchführung und Protokollierung Codereview (C1) und Validierung (D1) | 6.8 |

### 12.2.1 Analyse der Dokumentation

Als erster Schritt (so DIN EN ISO 13849-2) sei zu überprüfen, „*dass eine Dokumentation der Spezifikation und Gestaltung der sicherheitsbezogenen Software vorhanden ist. Diese Dokumentation ist zu untersuchen, um ihre Vollständigkeit sowie die Vermeidung von fehlerhaften Auslegungen, Unterlassungen und Widersprüchen zu überprüfen.*“

Dies ist also der Analyseschritt auf der Grundlage der Dokumentation und lässt sich bei vorhandener Excel-Datei zur Matrixmethode des IFA leicht durchführen. Die funktionalen Merkmale „Vollständigkeit“, „Korrektheit“ usw. sind durch die in SOFTEMA implementierten formalen Verifikationen umso schneller zu analysieren.

Davon abweichend merkt die Norm an:

„*ANMERKUNG Im Fall von kleinen Programmen kann eine Programmanalyse durch Nachprüfungen oder Analyse des Kontrollflusses (en: walk through), der Prozeduren usw. ausreichend sein, indem die Softwaredokumentation (Kontrollflussdiagramm, Quellcodes von Modulen oder Blöcken, I/O und Variablenzuweisungslisten, Querverweislisten) verwendet wird.*“

Für kleine Programme kann also anstelle der Dokumentation des Softwareentwurfs (Abbildung 6) eine Softwaredokumentation der Programmierumgebung genügen, wenn dabei auch der Kontrollfluss durch geeignete Diagramme nachvollziehbar ist. Die Eigenschaft „klein“ ist leider nicht spezifiziert.

### 12.2.2 Test der Software

Wenn die Analyse nicht ausreichend für die Validierung ist, dann erfolgen Black-Box-Tests[2] der SRASW auf der Zielhardware. In Abhängigkeit vom $PL_r$ und von der Kategorie sollten die Tests und deren Protokollierung die Anforderung aus Tabelle 25 umfassen. Muss nun jede bereits validierte SRASW erneut getestet werden? Dazu sagt die Norm: „*Einzelne Softwarefunktionen [Programmbausteine; Funktionsbausteine; Funktionen], die bereits validiert wurden, brauchen nicht erneut validiert zu werden. Wenn eine Anzahl derartiger Sicherheitsfunktionsblöcke für ein besonderes Projekt kombiniert wird, muss jedoch die sich daraus ergebende gesamte Sicherheitsfunktion validiert werden.*“

Auf die Matrixmethode des IFA umformuliert: Auch wenn alle Funktionsbausteine der Vorverarbeitungs- und der Ansteuerungsebene zertifiziert bzw. validiert sind, gilt es immer, die Verknüpfungslogik des Moduls ACT und die gesamte Sicherheitsfunktion des Programms zu validieren.

Sollte die sicherheitsbezogene Software nachträglich verändert werden, muss sie in angemessenem Umfang erneut validiert werden. Dies wird durch die Matrixmethode und SOFTEMA ebenfalls angemessen umgesetzt.

Tabelle 25:
Anforderungen an Tests und Protokollierung

| PL/Kategorie | Prüfmaßnahme nach DIN EN ISO13849-2 | Umsetzung in Matrixmethode des IFA (Dokumente) |
|---|---|---|
| alle $PL_r$ | Black-Box-Test des funktionellen Verhaltens und der Leistungsfähigkeit, z. B. Zeitverhalten | 1) Testplan der C&E-Matrix (B4) mit Zeilen für Sicherheitsfunktionen; mit Validierungsspalte und Name/Datum<br>2) Protokoll Validierung (D1) |
| empfohlen für $PL_r$ d oder e | zusätzlich erweiterte Testfälle, die auf Grenzwertanalysen beruhen | 1) Testplan der C&E-Matrix (B4) mit zusätzlichen Testfällen; mit Validierungsspalten und Name/Datum<br>2) Protokoll Validierung (D1) |
| alle $PL_r$ | I/O-Tests, um sicherzustellen, dass die sicherheitsbezogenen Eingangs- und Ausgangssignale richtig verwendet werden | 1) I/O-Liste (A2.4) mit Validierungsspalten und Name/Datum<br>2) Protokoll Validierung (D1) |
| $PL_r$ und Kategorien mit Fehlererkennung | Testfälle, die Fehler simulieren, die vorher analytisch bestimmt werden, zusammen mit der erwarteten Reaktion, um die Eignung der auf der Software beruhenden Maßnahmen zur Fehlerbeherrschung zu bewerten | 1) Testplan der C&E-Matrix (B4) mit zusätzlichen Testfällen; mit Validierungsspalten und Name/Datum<br>2) Protokoll Validierung (D1) |

## 12.3 Validierungsbeispiel aus DIN EN ISO 13849-2, Anhang E

Im Anhang E der DIN EN ISO 13849-2 [10] ist ein Beispiel zur Validierung eines SRP/CS angegeben. Die Softwarevalidierung ist dort aber nicht dargestellt. Daher präsentiert dieser Report für dieses Normenbeispiel eine mögliche Vorgehensweise anhand der Matrixmethode des IFA.

Das Besondere an diesem Beispiel ist, dass zwei Standard-SPSen eingesetzt werden. Entsprechend erfolgt die Spezifikation und Dokumentation zweifach: für jede SPS (PLC A und PLC B) in einer separaten Excel-Datei. Dabei ist die Spezifikation der Sicherheitsfunktionen (Dokument A1) identisch.

Die beiden Excel-Dokumente zu diesem Beispiel sind in einer separaten Archivdatei im Downloadbereich dieses Reports zu finden. Die Excel-Dateien können ebenfalls mit dem Tool SOFTEMA (Kapitel 14) geöffnet und betrachtet werden.

2 Black-Box-Test ist ein Test des dynamischen Verhaltens der Software unter realen funktionalen Bedingungen. Damit werden Abweichungen von der Softwarespezifikation aufgedeckt. Um den Test durchzuführen, wird kein Wissen über die interne Struktur der Software benötigt. Quelle: DIN EN 61508-7:2011 [3]

# 13 Technische Dokumentation und Benutzerinformation

DIN EN ISO 13849-1 gibt in den Kapiteln 10 und 11 Hinweise darauf, was zu dokumentieren und was davon für die Benutzung der Steuerung (im Allgemeinen an Betreiber der Maschine) weiterzugeben ist.

## 13.1 Technische Dokumentation

Bevor der Hersteller die EG-Konformitätserklärung für eine Maschine ausstellt, muss er eine technische Dokumentation ausarbeiten. In Bezug auf SRASW sind dazu im Normenkapitel 10 einige Stichworte relevant, die im Folgenden in Klammern referenziert werden. Zunächst sind die Spezifikation der realisierten Sicherheitsfunktionen mit den verschiedenen Gestaltungsdokumenten sowie das gut kommentierte Programm gefordert (*„Eigenschaften jeder Sicherheitsfunktion“, „Performance Level“, „Begründung der Gestaltung“, „Softwaredokumentation“*). Zusätzlich sind die benutzten zertifizierten oder selbst validierten Bibliotheksfunktionen mit ihrer Identifikation (Versionsnummer, Autor, Datum usw.) aufzulisten. Die Anwendung von Programmierrichtlinien und Sprachteilmengen ist ebenfalls zu dokumentieren (*„Maßnahmen gegen systematische Fehler“*). Falls das Programmierwerkzeug diese bereits enthält, genügt ein Hinweis auf diese Merkmale. Bleibt noch die Dokumentation der Testaktivitäten: Oft werden Integrationstest und Validierung der Sicherheitsfunktionen zusammen durchgeführt. Diese Tests sind selbstverständlich zu planen und mit Testergebnissen zu dokumentieren. Bei der Anwendung der Matrixmethode des IFA sind alle diese im Normenkapitel 10 geforderten Informationen für die Technische Dokumentation bereits vorhanden. Als Anmerkung heißt es: „*Im Allgemeinen ist diese Dokumentation für die herstellerinterne Verwendung gedacht und wird nicht an den Maschinennutzer weitergegeben*“. Dies sollte per Vertragsgestaltung anders geregelt werden, falls der Maschinenbetreiber selbst Softwaremodifikationen durchführen möchte oder muss und damit auf diese Dokumentation angewiesen ist.

Zur Softwaredokumentation gehört heutzutage auch ein Konfigurationsmanagement. Besonders bei sicherheitsbezogener Software ist verständlich und daher zu fordern, dass deren Entwicklung für alle Beteiligten und spätere Überprüfungen nachvollzogen werden kann, z. B.:

- Wer hat wann spezifiziert, programmiert, in Betrieb genommen, verifiziert, validiert?
- Womit wurde entwickelt, z. B. Werkzeuge und ihre Einstellungen, wiederverwendete Funktionen und ihre Identifikation, Programmierrichtlinie?
- Welche Programmversionen sind in welche Steuerungen geladen?

Diese und weitere notwendige Informationen sowie alle relevanten Entwicklungsdokumente sind für eine spätere Nutzung – z. B. bei einer Modifikation nach fünf Jahren Betrieb – zu dokumentieren und vor allem in geeigneter Weise zu archivieren. Gleichzeitig sollten alle verwendeten Werkzeuge in der genutzten Version archiviert werden, damit die Dokumente auch nach Jahren noch gelesen werden können.

## 13.2 Benutzerinformation

Was ist nach Kapitel 11 der DIN EN ISO 13849-1 als Benutzerinformation zu interpretieren? Es gibt keine speziellen, auf SRASW bezogenen Anforderungen. Es heißt allgemein: „*Insbesondere müssen die Informationen, die zur sicheren Verwendung der SRP/CS wichtig sind, dem Benutzer gegeben werden.*“

# 14 Das Softwaretool SOFTEMA zur Entwicklung und Prüfung von SRASW

Zur effizienten und qualitätsgesicherten Anwendung der Matrixmethode entwickelt das IFA ein Softwaretool namens SOFTEMA (Projektinformationsseite zum Projekt IFA5137: www.dguv.de/webcode/dp102081), das wie das IFA-Tool SISTEMA [15] zum freien Download verfügbar sein wird. Dieses Kapitel gibt einen Überblick über die grundlegenden Merkmale und Funktionen dieses Tools. Weitergehende Informationen und Benutzerhilfen werden separat auf der Downloadseite von SOFTEMA bereitgestellt (siehe Abschnitt 14.4).

## 14.1 Was kann SOFTEMA?

Mit SOFTEMA lassen sich die zum Download angebotenen Microsoft-Excel-Beispiele (Kapitel 7 und Abschnitt 12.3) betrachten. Darüber hinaus können mit SOFTEMA eigene Projekte neu erstellt und bearbeitet werden. SOFTEMA kann jeweils eine Projektdatei für die Spezifikation und Dokumentation eines Anwendungsprogrammes öffnen und bearbeiten. Die Software kann mehrfach ausgeführt werden, um verschiedene Projekte und Programme parallel betrachten und bearbeiten zu können. Somit können Projektdaten zwischen mehreren SOFTEMA-Instanzen (oder Excel-Instanzen) über die Zwischenablage kopiert und eingefügt werden.

SOFTEMA-Projektdateien verwenden den Dateityp „Microsoft-Excel-Arbeitsmappe (*.xlsx)“. Das Öffnen des früheren Excel-Dateityps *.xls ist dagegen nicht möglich, da dieser Dateityp u. a. nur 256 Tabellenspalten unterstützt. Die Projektdateien können wahlweise mit SOFTEMA, aber auch mit Microsoft Excel direkt bearbeitet werden. Mit Excel sind alle Tabellen frei editierbar, unter SOFTEMA sind die Inhalte durch die Benutzerverwaltung geschützt. Nur unter SOFTEMA sind dessen spezialisierte Funktionen, wie unten beschrieben, verfügbar. Die Projektdateien enthalten keine Makros. Alle SOFTEMA-Funktionen sind in die Software eingebunden und geschützt. Unter Excel können aber zusätzliche Tabellenblätter eingefügt und für die Entwicklung und Dokumentation genutzt werden, z. B. für die Dokumentation der Steuerungshardware. SOFTEMA ignoriert diese Tabellen und kann diese (in der ersten Freigabeversion) auch nicht laden oder anzeigen. Neben der Anwendung der Matrixmethode unterstützt SOFTEMA weitere Funktionen:

- automatische Aktualisierung von Tabellen bei Modifikation von Eingabedaten,
- formale Verifikation von Tabellen (auf fehlende, widersprüchliche oder doppelte Einträge),
- Verwaltung der Mitarbeitenden im Projekt,
- rollenbasierte Benutzerberechtigungen,
- Unterstützung bei der Verifikation, Validierung und Prüfung,
- Unterstützung bei Modifikationen,
- spezifische Editoren für die verschiedenen Zelleninhalte,
- Verwaltung von Dokumenten und Änderungen,
- Undo/Redo-Funktionen,
- Suchen/Ersetzen-Funktionen,
- spezifische Druckfunktionen und -reports,
- automatische Protokollierung von Änderungen besonders sicherheitskritischer Zelleninhalte.

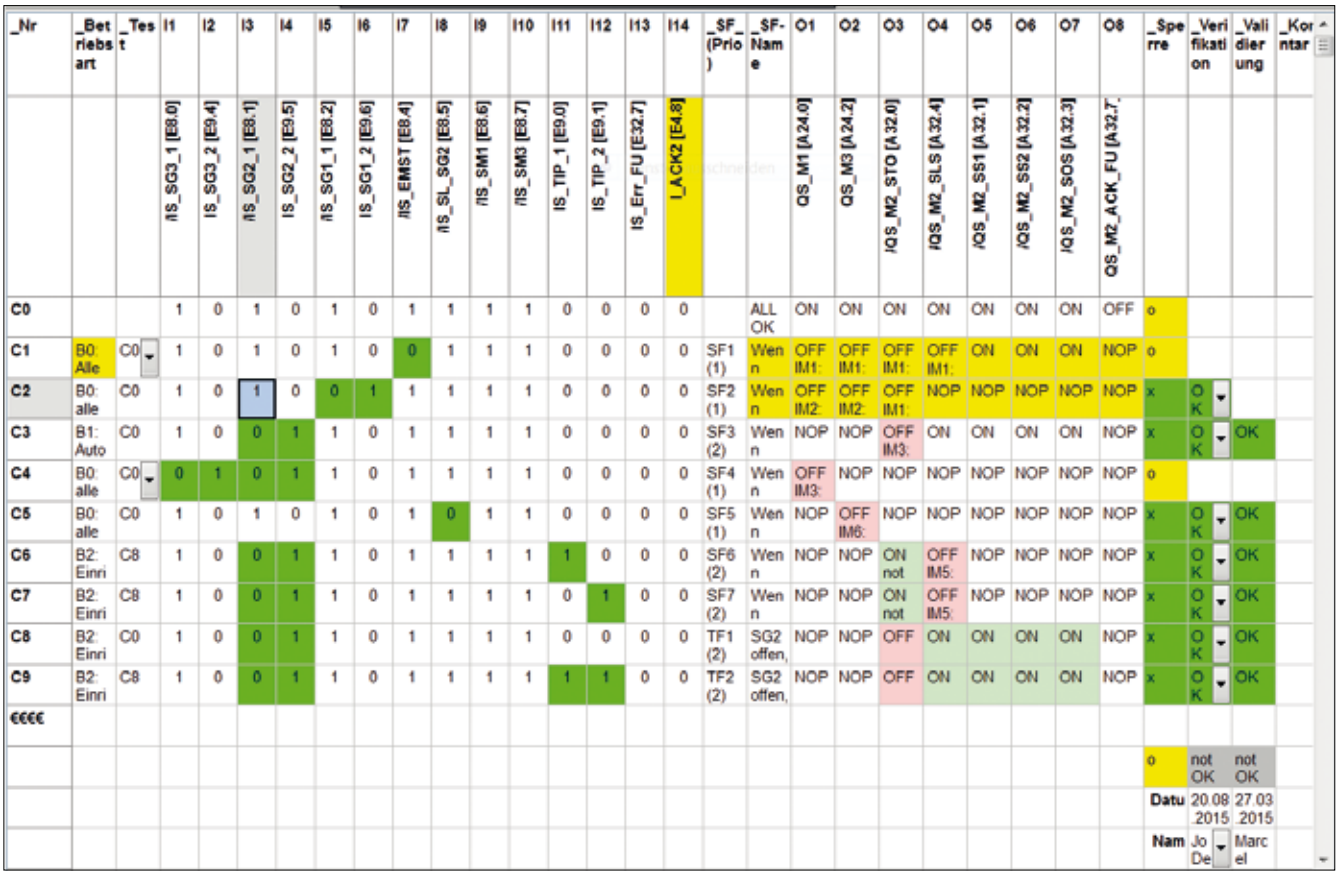
## 14.2 Wie wird SOFTEMA verwendet?

SOFTEMA verwaltet die für die Matrixmethode des IFA notwendigen Tabellen und darüber hinaus auch die für das Projektmanagement notwendigen Informationen wie Projektbeschreibung, Benutzerverwaltung, Änderungsprotokolle, Dokumentenmanagement usw. Abbildung 60 zeigt z. B. die C&E-Matrix eines Projektes in SOFTEMA.

Für ein neues Projekt eröffnet der Benutzer eine leere, aber schon vorformatierte Projektvorlage. Nach Ausfüllen der Projektbeschreibung (Tabelle Projekt) werden in Tabelle A1 „Sicherheitsfunktionen“ die Sicherheitsfunktionen mit ihren Eigenschaften wie $PL_r$, Betriebsart, Priorität usw. eingetragen. In Tabelle A2.4 „IO-Liste“ werden die Ein- und Ausgangssignale eingetragen, jeweils mit Variablennamen und Hardware/Netzwerk-Adressen. In alle Tabellen können auch externe Inhalte über die Zwischenablage kopiert und eingefügt werden. Der Katalog fehlervermeidender Maßnahmen und die Programmierregeln können in Tabelle A3 Maßnahmen ausgewählt und angepasst werden. Die Tabellen A3 „Maßnahmen“ und A4 „Anforderungen“ sollten schon vorab in der Projektvorlage vorbelegt sein. Anhand der Sicherheitsfunktionen, der Peripheriehardware und der I/O-Liste ergibt sich die Liste der erforderlichen Funktionsbausteine für Vorverarbeitungs- und Ansteuerungsebene. Diese sollten in Tabelle B3 „Modularchitektur“ verwaltet werden. Im Gegensatz zu den Darstellungen von Dokument B3 in diesem Report verwaltet SOFTEMA die Funktionsbausteine in einer Liste. Mit diesen Vorbereitungen kann die Tabelle B4 „Matrix C+E“ ausgefüllt werden. Dies erfolgt mit den Schaltflächen zur automatischen Aktualisierung für I/O-Signale und Sicherheitsfunktionen. Die eigentliche Softwarespezifikation erfolgt dann durch

- Zuordnen von Eingangssignalen zu den einzelnen Sicherheitsfunktionen und
- Eintragen der logischen Verknüpfung der Signale für die Schaltvorgänge auf die Ausgangssignale.

Abbildung 60:
C&E-Matrix in SOFTEMA



| _Nr | _Betriebsart | _Test | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 | I10 | I11 | I12 | I13 | I14 | _SF_(Prio) | _SF-Name | O1 | O2 | O3 | O4 | O5 | O6 | O7 | O8 | _Sperre | _Verifikation | _Validierung | _Kontar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | /IS_SG3_1 [E8.0] | IS_SG3_2 [E9.4] | /IS_SG2_1 [E8.1] | IS_SG2_2 [E9.5] | /IS_SG1_1 [E8.2] | IS_SG1_2 [E9.6] | /IS_EMST [E8.4] | /IS_SL_SG2 [E8.5] | /IS_SM1 [E8.6] | /IS_SM3 [E8.7] | IS_TIP_1 [E9.0] | IS_TIP_2 [E9.1] | IS_Err_FU [E32.7] | I_ACK2 [E4.8] | | | QS_M1 [A24.0] | QS_M3 [A24.2] | /QS_M2_STO [A32.0] | /QS_M2_SLS [A32.4] | /QS_M2_SS1 [A32.1] | /QS_M2_SS2 [A32.2] | /QS_M2_SOS [A32.3] | QS_M2_ACK_FU [A32.7] | | | | |
| C0 | | | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | ALL OK | ON | ON | ON | ON | ON | ON | ON | OFF | o | | | |
| C1 | B0: Alle | C0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | SF1 (1) | Wenn | OFF IM1: | OFF IM1: | OFF IM1: | OFF IM1: | ON | ON | ON | NOP | o | | | |
| C2 | B0: alle | C0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | SF2 (1) | Wenn | OFF IM2: | OFF IM2: | OFF IM1: | NOP | NOP | NOP | NOP | NOP | x | OK | | |
| C3 | B1: Auto | C0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | SF3 (2) | Wenn | NOP | NOP | OFF IM3: | ON | ON | ON | ON | NOP | x | OK | OK | |
| C4 | B0: alle | C0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | SF4 (1) | Wenn | OFF IM3: | NOP | NOP | NOP | NOP | NOP | NOP | NOP | o | | | |
| C5 | B0: alle | C0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | SF5 (1) | Wenn | NOP | OFF IM6: | NOP | NOP | NOP | NOP | NOP | NOP | x | OK | OK | |
| C6 | B2: Einri | C8 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | SF6 (2) | Wenn | NOP | NOP | ON not | OFF IM5: | NOP | NOP | NOP | NOP | x | OK | OK | |
| C7 | B2: Einri | C8 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | SF7 (2) | Wenn | NOP | NOP | ON not | OFF IM5: | NOP | NOP | NOP | NOP | x | OK | OK | |
| C8 | B2: Einri | C0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | TF1 (2) | SG2 offen, | NOP | NOP | OFF | ON | ON | ON | ON | NOP | x | OK | OK | |
| C9 | B2: Einri | C8 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | TF2 (2) | SG2 offen, | NOP | NOP | OFF | ON | ON | ON | ON | NOP | x | OK | OK | |
| €€€€ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | o | not OK | not OK | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | Datu | 20.08.2015 | 27.03.2015 | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | Nam | Jo De | Marcel | |

Letzteres wird für die Codierung der Ansteuerlogik benötigt. Ein spezialisierter Editor hilft bei dieser Verknüpfung. Bei umfangreichen Projekten hilft die kompakte Darstellung in Tabelle B4 Matrix kompakt. Man erstellt diese Tabelle allein durch die Aktualisierungsfunktion, die die Tabelle B4 „Matrix C+E“ automatisch umwandelt. Spätestens zu diesem Zeitpunkt sollten alle verfügbaren Funktionen zur formalen Verifikation der genannten Tabellen genutzt worden sein, um Auslassungen, Dubletten und Widersprüche aufdecken und korrigieren zu können.

Nach der Verifikation aller Eingangsdokumente und der oben beschriebenen Spezifikation kann die Codierung des Programms erfolgen. Der Code wird ebenfalls verifiziert. Dieser Vorgang wird in verschiedenen Tabellen im Detail und zusammenfassend auch in C1 „Codereview“ dokumentiert. Danach wird das Programm validiert, was ebenfalls in verschiedenen Tabellen einzeln dokumentiert und in Tabelle D1 „Validierung“ zusammengefasst wird. In den Tabellen C1 und D1 können die Fragen nach Bedarf angepasst und auch ergänzt werden. Personen, die anschließend das Projekt prüfen, können ihre Tätigkeit ebenfalls dokumentieren und kommentieren.

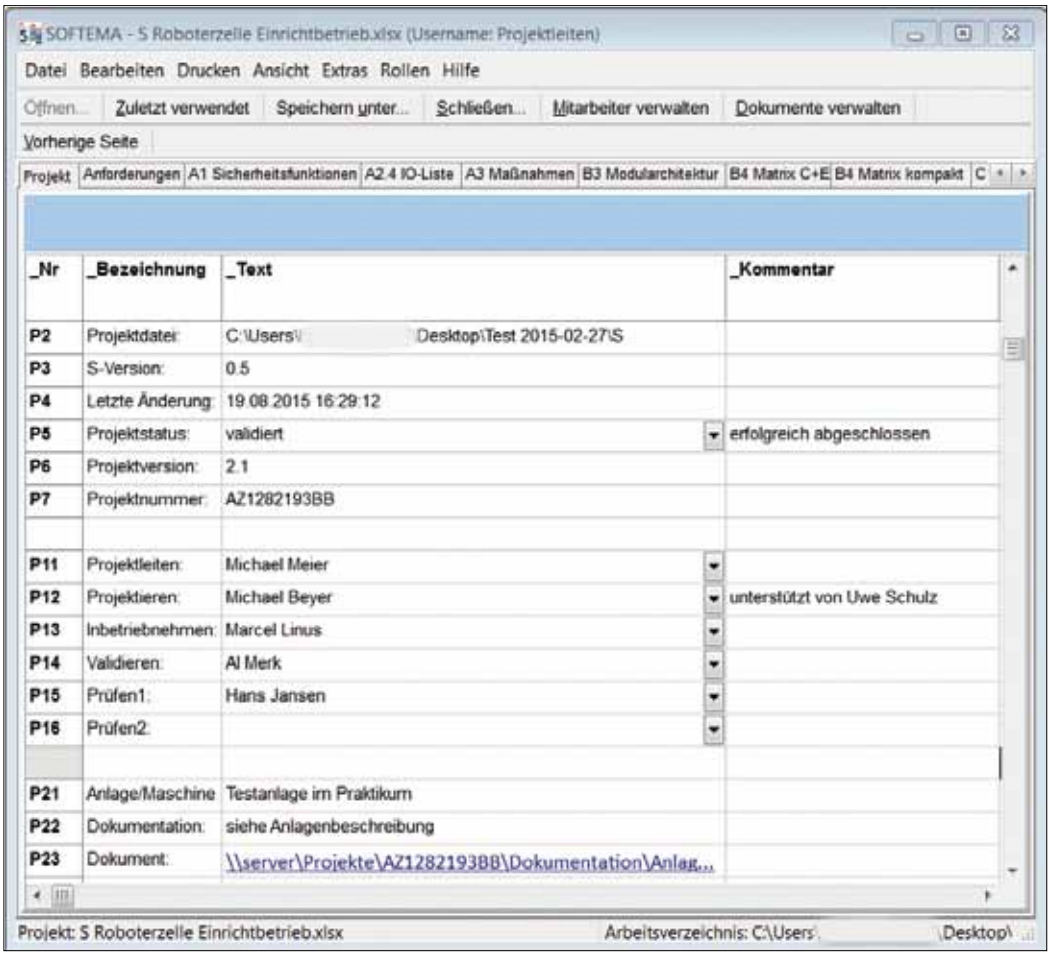
Bei Modifikationen der Sicherheitsfunktionen oder der I/O-Signale werden die Änderungen aus den Tabellen A1 und A2.4 wiederum in den Spezifikationstabellen automatisch aktualisiert und vom Benutzer überarbeitet. Alle Modifikationen werden zunächst farblich (gelb) markiert. Die Markierungen werden nach Abschluss der erneuten Codierung, Verifikation und Validierung dieser Modifikationen manuell gelöscht.

## 14.3 Die Benutzerschnittstelle von SOFTEMA

SOFTEMA ist eine Applikation für die Microsoft-Betriebssysteme Windows 7, Windows 8 oder Windows 10. Es nutzt die klassische Menütechnik mit einer festen Symbolleiste für die wichtigsten Befehle. In der Titelleiste (oben) und in der Statuszeile (unten) werden weitere Informationen angezeigt.

Die Programmoberfläche von SOFTEMA (Abbildung 61) zeichnet sich durch intuitive Bedienbarkeit aus, weil sie sich stark an den Konzepten und der Bedienung von MS Excel orientiert. Den größten Anteil der Programmoberfläche nimmt der Arbeitsbereich, eine Tabelle, in der Mitte ein (siehe Abbildung 61). SOFTEMA verwaltet alle Eingaben in Tabellen, die über die Registerkarten am oberen Rand des Arbeitsbereiches angewählt werden können. Jede Tabelle entspricht einem Tabellenblatt in der Excel-Datei. In jeder Registerkarte befindet sich über der eigentlichen Tabelle noch ein Bereich mit tabellenbezogenen Funktionen, die über Schaltflächen, Auswahllisten oder Optionsfelder bedient werden können.

Abbildung 61:
Programmoberfläche von SOFTEMA, Darstellung der Projektbeschreibung




## 14.4 Wo ist SOFTEMA zu erhalten?

Das Tool SOFTEMA wird im Jahr 2017 auf den Internetseiten des IFA nach Registrierung als Freeware zur kostenlosen Benutzung angeboten. Aktuelle Informationen über den Entwicklungsstand, Betaversionen sowie den Link zum Download sind unter der Internetadresse www.dguv.de/ifa, Webcode d1082520 erhältlich. Beachten Sie dabei den Haftungsausschluss und die lizenzrechtlichen Hinweise.

## 14.5 Wie wird SOFTEMA installiert und ausgeführt?

SOFTEMA wird mit dem mitgelieferten Installationsprogramm installiert. Zunächst wird nur eine deutsche Sprachversion angeboten. Obwohl die Projektdateien den Microsoft-Excel-Dateityp *.xlsx verwenden, ist eine Installation von Microsoft Excel nicht notwendig – aber für eine gelegentliche direkte Bearbeitung der Projektdateien durchaus sinnvoll.

# 15 Literatur


[1] DIN EN ISO 13849-1: Sicherheit von Maschinen – Sicherheitsbezogene Teile von Steuerungen – Teil 1: Allgemeine Gestaltungsleitsätze (12/2008); sowie Änderung 1 der DIN EN ISO 13849-1 (erscheint 2016). Beuth, Berlin 2008/2016

[2] *Hauke, M.; Schaefer, M.; Apfeld, R.; Bömer, T.; Huelke, M.* et al.: Funktionale Sicherheit von Maschinensteuerungen – Anwendung der DIN EN ISO 13849 (BGIA-Report 2/2008). Hrsg.: Deutsche Gesetzliche Unfallversicherung (DGUV), Berlin 2008 (derzeit in Überarbeitung)

[3] DIN EN 61508: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/ programmierbarer elektronischer Systeme – Teil 1 bis Teil 7 (alle 02/2011). Beuth, Berlin 2011

[4] DIN EN 954-1: Sicherheit von Maschinen – Sicherheitsbezogene Teile von Steuerungen – Teil 1: Allgemeine Gestaltungsleitsätze (3/1997). Beuth, Berlin 1997 (nicht mehr gültig)

[5] Normgerechte Entwicklung und Dokumentation von sicherheitsbezogener Anwendersoftware im Maschinenbau. Projekt Nr. FF-FP0319. Hrsg.: Deutsche Gesetzliche Unfallversicherung (DGUV), Berlin 2014. www.dguv.de, Webcode dp54444

[6] DIN EN 62061: Sicherheit von Maschinen – Funktionale Sicherheit sicherheitsbezogener elektrischer, elektronischer und programmierbarer elektronischer Steuerungssysteme (09/2013). Beuth, Berlin 2013

[7] VDI/VDE-Richtlinie: Applikationsprogrammierung von Sicherheitsfunktionen für Maschinenbau und Fertigungstechnik (Entwurf). GMA AK1.50 Methoden der Steuerungstechnik

[8] PLCopen – Technical Committee 5 – Safety Software Technical Specification, Part 1: Concepts and Function Blocks Version 1.0 – Official Release, 2006

[9] *Becker, N.; Eggeling, M.; Huelke, M.*: SPS-Software für fehlersichere Steuerungen – Normgerecht entwickeln und dokumentieren. atp edition – Automatisierungstechnische Praxis 57 (2015) Nr. 4, S. 34-47

[10] DIN EN ISO 13849-2: Sicherheit von Maschinen – Sicherheitsbezogene Teile von Steuerungen – Teil 2: Validierung (2/2013). Beuth, Berlin 2013

[11] DIN ISO/TR 23849: Leitfaden zur Anwendung von ISO 13849-1 und IEC 62061 bei der Gestaltung von sicherheitsbezogenen Steuerungen für Maschinen (12/2014). Beuth, Berlin 2014

[12] DIN EN 61131-3: Speicherprogrammierbare Steuerungen – Teil 3: Programmiersprachen (IEC 61131-3:2013) (06/2014). Beuth, Berlin 2014

[13] DIN EN ISO 12100: Sicherheit von Maschinen – Allgemeine Gestaltungsleitsätze – Risikobeurteilung und Risikominderung (3/2011). Beuth, Berlin 2011

[14] Definition von Sicherheitsfunktionen – Was ist wichtig? (SISTEMA-Kochbuch 6). Hrsg.: Deutsche Gesetzliche Unfallversicherung (DGUV), Berlin 2015. www.dguv.de/ifa, Webcode d109240

[15] Software-Assistent SISTEMA. Hrsg.: Deutsche Gesetzliche Unfallversicherung (DGUV), Berlin. www.dguv.de/ifa, Webcode d11223

[16] *Boehm, B.*: Guidelines for verifying and validating software requirements and design specifications. In: *Samet, P. A.* (Hrsg.): Euro IFIP: Proceedings of the European Conference on Applied Information Technology of the International Federation for Information Processing, London, 25.-28. September 1979. S. 711-719. North-Holland, Amsterdam 1979

[17] *Becker, N.; Eggeling, M.*: Abschlussbericht zum DGUV-Projekt Nr. FF-FP0319. Hrsg.: Deutsche Gesetzliche Unfallversicherung (DGUV), Berlin 2013. www.dguv.de, Webcode dp54444

[18] DIN EN 61508-3: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme – Teil 3: Anforderungen an Software (2/2011). Beuth, Berlin 2011

[19] *Barg, J.; Eisenhut-Fuchsberger, F.; Orth, A.; Ost, J.; Springhorn, C.*: 10 Schritte zum Performance Level: Handbuch zur Umsetzung der funktionalen Sicherheit nach ISO 13849. Hrsg.: Bosch Rexroth, Würzburg 2011

[20] *Bömer, T.; Schaefer, M.*: Unterschiede bei der Verwendung von fertigen Sicherheitsbauteilen und Standardbauteilen für die Realisierung von Sicherheitsfunktionen an Maschinen. Hrsg.: Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA), Sankt Augustin 2011

[21] Wenn die vorgesehenen Architekturen nicht passen (SISTEMA-Kochbuch 4). Hrsg.: Deutsche Gesetzliche Unfallversicherung (DGUV), Berlin 2012. www.dguv.de/ifa, Webcode d109240

[22] *Mai, M.; Reuß, G.*: Selbsttests für Mikroprozessoren mit Sicherheitsaufgaben oder: Quo vadis Fehler? (BGIA-Report 7/2006). Hrsg.: Hauptverband der gewerblichen Berufsgenossenschaften (HVBG), Sankt Augustin 2006. www.dguv.de/ifa, Webcode d6163

[23] *Ostermann, B.*: Entwickeln und Bewerten Fehler erkennender Programmbausteine in speicherprogrammierbaren Steuerungen (SPS) zur Erhöhung deren Sicherheit. Diplomarbeit. Fachhochschule Bonn-Rhein-Sieg, Sankt Augustin 2006.
www.maschinenbautage.eu/index.php?id=289

[24] *Gall, H.; Kemp, K.*: Wirksamkeit von zeitlichen und logischen Programmlaufüberwachungen beim Betrieb von Rechnersystemen. – Grundlagen der Rechnersicherheit – Leitlinien und Programm für den Einsatz von Programmlaufüberwachungen. Schriftenreihe der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin: Forschungsbericht, Fb 772. Hrsg: Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA), Dortmund 1997. Bremerhaven, Wirtschaftsverlag NW Verlag für neue Wissenschaft 1997

# 16 Abkürzungsverzeichnis

| Abkürzung | Bezeichnung |
|---|---|
| ACT | Bezeichnung der Ansteuerlogik; von actuate = ansteuern |
| BGIA | Berufsgenossenschaftliches Institut für Arbeitsschutz (heute: IFA) |
| CAE | Computer Aided Engineering |
| C&E-Matrix | Cause and Effect Matrix; synonym: Cause and Effect Table; Ursache-Wirkungs-Diagramm |
| CPU | Central Processing Unit |
| $DC/DC_{avg}$ | Diagnostic coverage; Diagnosedeckungsgrad/average diagnostic coverage; mittlerer Diagnosedeckungsgrad |
| EMV | Elektromagnetische Verträglichkeit |
| FBD | SPS-Sprache: Function block diagram [18], Funktionsbausteinsprache |
| FMEA | Failure mode and effect analysis; Ausfalleffektanalyse |
| FU | Frequenzumrichter |
| FVL | Full variability language; Programmiersprache mit nicht eingeschränktem Sprachumfang |
| IFA | Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung |
| I/O | Input/Output; Eingang/Ausgang einer SPS |
| KAN | Kommission Arbeitschutz und Normung |
| KOP/LD | SPS-Sprache: Kontaktplan; englisch: Ladder diagram (LD) |
| LVL | Limited variability language; Programmiersprache mit eingeschränktem Sprachumfang |
| $MTTF_D$ | Mean time to dangerous failure; mittlere Zeit bis zum gefahrbringenden Ausfall |
| NOP | No operation; Nulloperation: Befehl in der C&E-Matrix, der nichts bewirkt |
| $PFH_D$ | Probability of a dangerous failure per hour; Wahrscheinlichkeit eines gefährlichen Ausfalls pro Stunde |
| PL | Performance Level |
| $PL_r$ | Required Performance Level; erforderlicher Performance Level |
| PLC | Programmable Logic Controller; Speicherprogrammierbare Steuerung |
| SF | Safety function; Sicherheitsfunktion |
| SIL | Safety integrity level |
| SISTEMA | Softwareassistent des IFA „Sicherheit von Steuerungen an Maschinen“ |
| SOFTEMA | Softwareassistent des IFA „Sichere Software an Maschinen“ |
| SPS(en) | Speicherprogrammierbare Steuerung(en) |
| SRASW | Safety-related application software; sicherheitsbezogene Anwender-Software |
| SRESW | Safety-related embedded software; sicherheitsbezogene eingebettete Software |
| SRP/CS | Safety related parts of control systems; sicherheitsbezogene Teile von Steuerungen |
| SSPS(en) | Speicherprogrammierbare Safety-Steuerung(en) |
| TÜV | Technischer Überwachungsverein |
| VDMA | Verband Deutscher Maschinen- und Anlagenbauer |

**Deutsche Gesetzliche**
**Unfallversicherung e.V. (DGUV)**

**Glinkastr. 40**
**10117 Berlin**
**Telefon: 030 288763800**
**Fax: 030 288763808**
**E-Mail: info@dguv.de**
**Internet: www.dguv.de**