(19) Deutsches Patent- und Markenamt

(10) **DE 10 2014 109 569 A1** 2015.01.15

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2014 109 569.3**
(22) Anmeldetag: **09.07.2014**
(43) Offenlegungstag: **15.01.2015**

(51) Int Cl.: ***G05B 19/045*** (2006.01)

(30) Unionspriorität:
13/937,805 **09.07.2013** **US**

(71) Anmelder:
Fisher-Rosemount Systems, Inc., Round Rock, Tex., US

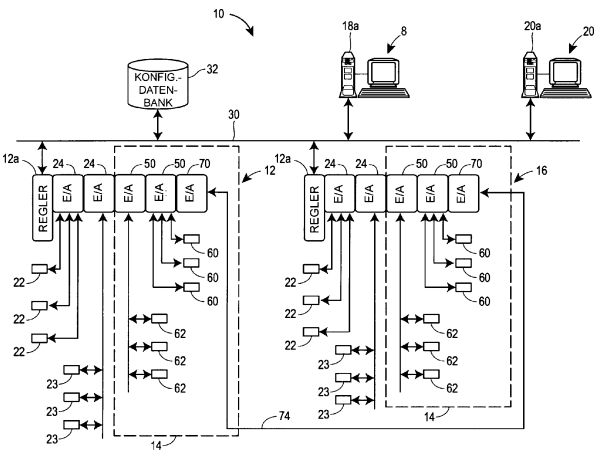
(74) Vertreter:
Meissner, Bolte & Partner GbR, 80538 München, DE

(72) Erfinder:
Law, Gary Keith, Georgetown, Tex., US; Sherriff, Godfrey R., Austin, Tex., US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **ZUSTANDSMASCHINEN-FUNKTIONSBLOCK MIT BENUTZERDEFINIERBAREN AKTIONEN AN EINEM ÜBERGANG ZWISCHEN ZUSTÄNDEN**

(57) Zusammenfassung: Ein Regelungssystem, ein Sicherheitssystem usw. innerhalb einer Prozessanlage können jeweils einen oder mehrere Zustandsmaschinen-Funktionsblöcke verwenden, die mühelos in eine Programmierumgebung mit Funktionsblockdiagramm zu integrieren ist. Ein derartiger Zustandsmaschinen-Funktionsblock kann eine oder mehrere Eingaben umfassen, die bewirken können, dass eine Zustandsmaschine, die durch den Zustandsmaschinen-Funktionsblock umgesetzt wird, einen nächsten Zustand sowie eine oder mehrere Übergangsaktionen, die gemäß dem Übergang von einem aktuellen. Zustand in den nächsten Zustand auszuführen ist bzw. sind, identifiziert. Konfigurationsdaten, die mit den Übergangsaktionen verknüpft sind, können basierend auf dem aktuellen und dem nächsten Zustand der Zustandsmaschine und mindestens einer der Eingaben aus einer Datenbank abgerufen werden. Der Zustandsmaschinen-Funktionsblock kann auch eine oder mehrere Ausgaben umfassen, die basierend auf dem Zustandsübergang generiert werden.



Beschreibung

GEBIET DER ERFINDUNG

[0001] Die vorliegende Offenbarung betrifft im Allgemeinen Funktionsblöcke zur Verwendung in Prozessanlagen und genauer gesagt das Konfigurieren und Umsetzen einer Zustandsmaschine, die zu einer Prozessanlage gehört.

ALLGEMEINER STAND DER TECHNIK

[0002] Prozessregelungssysteme, wie sie bei Chemie-, Erdöl- oder anderen Prozessen verwendet werden, umfassen typischerweise einen oder mehrere Prozessregler, die mit mindestens einer Host- oder Bedienerarbeitsstation und mit einem oder mehreren Feldgeräten über analoge, digitale oder kombinierte Analog/Digital-Busse oder Leitungen kommunikationsmäßig gekoppelt sind. Die Feldgeräte, bei denen es sich beispielsweise um Ventile, Ventilpositionierer, Schalter, Transmitter (z. B. Temperatur-, Druck- und Durchsatzsensoren) handeln kann, erfüllen Funktionen innerhalb der Prozessanlage, wie etwa das Öffnen oder Schließen von Ventilen und das Messen von Prozessparametern. Die Prozessregler empfangen Signale, die Prozessmessungen, die von den Feldgeräten erstellt werden, und/oder andere Informationen, die sich auf die Feldgeräte beziehen, angeben, verwenden diese Informationen, um Regelungsroutinen umzusetzen, und generieren dann Regelsignale, die über die Busse oder Leitungen an die Feldgeräte gesendet werden, um den Betrieb des Prozesses zu regeln. Die Informationen von den Feldgeräten und den Reglern werden typischerweise einer oder mehreren Anwendungen zur Verfügung gestellt, die von der Bedienerarbeitsstation ausgeführt werden, um es einem Bediener zu ermöglichen, eine beliebige gewünschte Funktion mit Bezug auf den Prozess auszuführen, wie etwa das Konfigurieren des Prozesses, das Visualisieren des derzeitigen Zustands des Prozesses, das Ändern der Funktionsweise des Prozesses usw.

[0003] Zusätzlich wird in vielen Prozessen ein separates Sicherheitssystem bereitgestellt, um bedeutende sicherheitstechnische Probleme innerhalb der Prozessanlage zu ermitteln und automatisch Ventile zu schließen, Energie zu den Geräten abzuschalten, Strömungen innerhalb der Anlage umzuschalten usw., wenn ein Problem auftritt, das eine ernste Gefahr in der Anlage ergeben oder dazu führen könnte, wie etwa ein Überlaufen giftiger Chemikalien, eine Explosion usw. Diese Sicherheitssysteme verfügen typischerweise über einen oder mehrere separate Regler außer den standardmäßigen Reglern zur Prozessregelung, die genannten Logic Solver, die an Sicherheitsfeldgeräte über separate Busse oder Kommunikationsleitungen, die innerhalb der Prozessanlage installiert sind, angeschlossen sind. Die Logic Solver verwenden die Sicherheitsfeldgeräte, um Prozessbedingungen zu ermitteln, die mit bedeutenden Ereignissen verknüpft sind, wie etwa mit der Position von bestimmten Sicherheitsschaltern oder Abschaltventilen, Überläufen oder Unterläufen im Prozess, dem Betrieb wichtiger Energieerzeugungs- oder Regelgeräten, dem Betrieb von Fehlerdetektionsvorrichtungen usw., um dadurch „Ereignisse" innerhalb der Prozessanlage zu ermitteln. Wenn ein Ereignis (typischerweise als „Ursache" bezeichnet) ermittelt wird, trifft der Sicherheitsregler gewisse Maßnahmen (typischerweise als „Effekt" bezeichnet), um die schädliche Beschaffenheit des Ereignisses einzuschränken, wie etwa durch Schließen von Ventilen, Ausschalten von Vorrichtungen, Abschalten von Energie für Partien der Anlage usw. Im Allgemeinen umfassen diese Maßnahmen oder Effekte das Schalten von Sicherheitsvorrichtungen in einen ausgelösten oder „sicheren" Betriebsmodus, der ausgelegt ist, um einen ernsten oder gefährlichen Zustand innerhalb der Prozessanlage zu verhindern.

[0004] Systeme innerhalb einer Prozessanlage, wie etwa Prozessregelungssysteme und Sicherheitssysteme, können typischerweise die Zustände diverser Prozesse und/oder der Systeme selber verfolgen. Eingangssignale in ein System können bewirken, dass sich der Status, der von dem System verfolgt wird, ändert, und Ausgabesignale, die von dem System generiert werden, können von dem aktuellen Status des Systems zusätzlich zu den Eingabesignalen für das System abhängen. Das US-Patent Nr. 7,730,415, das hiermit zur Bezugnahme vollständig übernommen wird, beschreibt ein Regelungssystem innerhalb einer Prozessanlage, das Zustandsmaschinen-Funktionsblöcke verwendet, die in eine Programmierumgebung mit Funktionsblockdiagramm integriert sind. Insbesondere umfasst ein derartiger Zustandsmaschinen-Funktionsblock eine oder mehrere Eingaben, die verwendet werden, um zu bewirken, dass sich der Zustand einer Zustandsmaschine, die von dem Zustandsmaschinen-Funktionsblock umgesetzt wird, ändert. Ferner bestimmt der Zustandsmaschinen-Funktionsblock einen nächsten Zustand, in den er übergehen soll, basierend auf Zustandsübergangs-Konfigurationsdaten, die den nächsten Zustand angeben. Die Zustandsübergangs-Konfigurationsdaten werden basierend auf dem derzeitigen Zustand der Zustandsmaschine und mindestens einer der Eingaben aus einer Datenbank abgerufen. Der Zustandsmaschinen-Funktionsblock umfasst auch eine oder mehrere Ausgaben, die basierend auf dem Zustand der Zustandsmaschine generiert wird bzw. werden. Die Eingaben des Zustandsmaschinen-Funktionsblocks sind beispielsweise mit einem Prozessregelungssystem oder einem Sicherheitssystem verknüpft, und die Ausgaben können beispielsweise zur Regelung von Feldgeräten in dem Prozessregelungssystem oder dem Sicherheitssystem verwendet werden.

[0005] Die derzeitigen Prozessregelungssysteme sind jedoch nicht in der Lage, diverse Aktionen oder Funktionen automatisch auszuführen, die mit einem Übergang aus einem aktuellen Zustand oder einem Übergang in einen nächsten Zustand verknüpft sind. Stattdessen müssen die Benutzer und Administratoren von derzeitigen Prozessregelungssystemen die Aktionen oder Funktionen während der Zustandsübergänge manuell ausführen oder umsetzen. Daher sind die derzeitigen Prozessregelungssysteme in ihrer Fähigkeit, gewisse Sicherheitsmaßnahmen, Regeltechniken und andere Merkmale auszuführen, die mit Zustandsübergängen verknüpft sind, eingeschränkt.

KURZDARSTELLUNG

[0006] Die Systeme und Verfahren, wie sie hier beschrieben werden, betreffen die Handhabung von Übergängen zwischen den Zuständen einer Prozessregelumgebung. Die Systeme und Verfahren zur Prozessregelung können eine Zustandsmaschine umsetzen, die eine Übergangstabelle umfasst, die Übergänge zwischen diversen Zuständen der Zustandsmaschine basierend auf bestätigten Eingaben identifiziert. Die Übergangstabelle kann ferner eine oder mehrere Übergangsaktionen für die Systeme und Verfahren zur Prozessregelung vorgeben, die in Zusammenhang mit dem Übergang von einem aktuellen Zustand in einen nächsten Zustand auszuführen sind. Gemäß den Ausführungsformen können die Übergangsaktionen in Form von Übergang-in-Aktionen und/oder Übergang-aus-Aktionen vorliegen. Die Systeme und Verfahren zur Prozessregelung können die Übergang-aus-Aktionen in Zusammenhang mit dem Übergang aus einem aktuellen Zustand ausführen und können die Übergang-in-Aktionen vor dem Stabilisieren in einen nächsten Zustand ausführen. Bei einigen Ausführungsformen können die Systeme und Verfahren zur Prozessregelung Konfigurationsdaten, die mit den Übergangsaktionen verknüpft sind, einem anderen Funktionsblock zur Ausführung durch diesen Funktionsblock bereitstellen. Die Systeme und Verfahren zur Prozessregelung können zusätzlich Ausgaben einstellen, die dem derzeitigen Zustand und den Übergangsaktionen entsprechen.

[0007] Gemäß den Ausführungsformen können die Systeme und Verfahren zur Prozessregelung die Übergangstabelle über Matrizen handhaben, die von einer grafischen Benutzerschnittstelle angezeigt werden können. Diverse Zellen der Matrizen können Zustandsübergangsdaten angeben, die diverse Zustandsübergänge, die mit bestätigten Eingaben verknüpft sind, zusammen mit Übergangsaktionen, die in Zusammenhang mit den Zustandsübergängen auszuführen sind, identifizieren. Die Matrizen können über einen Computer oder seinen Benutzer voll konfigurierbar sein, um Zustandsübergänge und Übergangsaktionen, die damit verknüpft sind, vorzugeben. Daher kann ein Funktionsblock auf die geeigneten Matrizen zugreifen, um Zustandsübergänge zu ermöglichen, automatisch die verknüpften Übergangsaktionen auszuführen und geeignete Ausgaben einzustellen.

[0008] Die Ausführungsformen der hier beschriebenen Systeme und Verfahren zur Prozessregelung können im Vergleich zu herkömmlichen Prozessregeltechniken zu wirksameren und effizienteren Prozessregeltechniken führen. Beispielsweise ermöglichen es die Übergangsaktionen den Systemen und Verfahren zur Prozessregelung, automatisch Aktionen auszuführen, die mit dem Übergang aus einem derzeitigen Zustand und dem Übergang in einen nächsten Zustand verknüpft sind, wodurch die Notwendigkeit vermindert wird, dass die Benutzer Prozesse, die mit Zustandsübergängen verknüpft sind, manuell ausführen müssen.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0009] Die Merkmale und Vorteile der hier beschriebenen Verfahren, Geräte und Systeme werden mit Bezug auf die folgende ausführliche Beschreibung und die beiliegenden Zeichnungen am besten verständlich werden. Es zeigen:

[0010] **Fig.** 1 ein Blockdiagramm einer beispielhaften Prozessanlage;

[0011] **Fig.** 2 ein Blockdiagramm einer beispielhaften Arbeitsstation, die in **Fig.** 1 schematisch abgebildet ist;

[0012] **Fig.** 3 eine beispielhafte Anzeige, die ein Regelmodul darstellt;

[0013] **Fig.** 4 ein Beispiel einer Darstellung eines Zustandsmaschinen-Funktionsblocks;

[0014] **Fig.** 5 eine beispielhafte Matrix zum Eingeben von Zustandskonfigurationsdaten und Übergangsaktionsdaten, die damit für einen Zustandsmaschinen-Funktionsblock verknüpft sind;

[0015] **Fig.** 6 die beispielhafte Matrix aus **Fig.** 5, bei der Zustandskonfigurationsdaten und Übergangsaktionsdaten, die damit verknüpft sind, in der Matrix angezeigt werden;

[0016] **Fig.** 7 ein Ablaufschema eines beispielhaften Verfahrens der Wirkungsweise eines Zustandsmaschinen-Funktionsblocks mit Übergangsaktionsfunktion;

[0017] **Fig.** 8 ein Blockdiagramm eines beispielhaften Zustandsmaschinen-Funktionsblocks, der eine Übergangsaktionsfunktion integriert;

[0018] Fig. 9 ein Ablaufschema eines anderen beispielhaften Verfahrens der Wirkungsweise eines Zustandsmaschinen-Funktionsblocks mit Übergangsaktionsfunktion;

[0019] Fig. 10 ein Ablaufschema einer beispielhaften Routine zum Verarbeiten von Dateneingaben in einen Zustandsmaschinen-Funktionsblock;

[0020] Fig. 11 ein Ablaufschema einer beispielhaften Routine zum Verarbeiten einer Freigabe-Eingabe für einen Zustandsmaschinen-Funktionsblock;

[0021] Fig. 12 ein Ablaufschema einer beispielhaften Routine zum Ändern eines Zustands und zum Einstellen von Ausgaben, wozu Übergangsaktionsausgaben eines Zustandsmaschinen-Funktionsblocks gehören;

[0022] Fig. 13 eine beispielhafte Matrix zum Eingeben von Ausgabekonfigurationsdaten, wozu Übergangsaktionen für einen Zustandsmaschinen-Funktionsblock gehören;

[0023] Fig. 14 ein Blockdiagramm eines anderen beispielhaften Zustandsmaschinen-Funktionsblocks; und

[0024] Fig. 15 ein beispielhaftes Zustandsübergangsdiagramm zum Eingeben von Zustandskonfigurationsdaten und Übergangsaktionsdaten, die damit verknüpft sind, für einen Zustandsmaschinen-Funktionsblock.

AUSFÜHRLICHE BESCHREIBUNG

Beispielhafte Prozessanlage

[0025] Fig. 1 ist ein Blockdiagramm einer beispielhaften Prozessanlage **10**, die ein oder mehrere Knoten **12**, **16**, **18** und **20** umfasst. In der beispielhaften Prozessanlage **10** aus **Fig.** 1 umfasst jeder der Knoten **12** und **16** einen Prozessregler **12a**, **16a**, der an ein oder mehrere Feldgeräte **22** und **23** über Ein-/Ausgabe-(E/A)Geräte **24** angeschlossen ist, die beispielsweise Foundation-Fieldbus-Schnittstellen, HART-Schnittstellen usw. sein können. Die Regler **12a** und **16a** sind auch mit einer oder mehreren Host- oder Bedienerarbeitsstationen **18a** und **20a** in dem Knoten **18** und **20** über ein Netzwerk **30**, das beispielsweise einen oder mehrere von einem Bus, einem drahtgebundenen lokalen Netzwerk (LAN), wie etwa einem Ethernet-LAN, einem drahtlosen LAN, einem Großraumnetzwerk (WAN), einem Intranet usw. umfassen kann, gekoppelt. Während sich die Reglerknoten **12**, **16** und die E/A-Geräte **24** und die damit verknüpften Feldgeräte **22**, **23** typischerweise innerhalb der manchmal rauen Anlagenumgebung befinden und darin verteilt sind, befinden sich die Knoten **18** und **20** der Bedienerarbeitsstation gewöhnlich in Regelwarten oder anderen weniger rauen Umgebungen, die für das Reglerpersonal leicht zugänglich sind.

[0026] Im Allgemeinen können die Arbeitsstationen **18a** und **20a** der Knoten **18** und **20** verwendet werden, um Anwendungen zu speichern und auszuführen, die verwendet werden, um die Prozessanlage **10** zu konfigurieren und zu überwachen, und/oder um die Geräte **22**, **23**, **24** und Regler **12a**, **16a** in der Prozessanlage **10** zu verwalten. Ferner kann eine Datenbank **32** an das Netzwerk **30** angeschlossen sein und als Datenarchiv und/oder Konfigurationsdatenbank funktionieren, welche die aktuelle Konfiguration der Prozessanlage **10** speichert, wie sie innerhalb der Knoten **12**, **16**, **18**, **20**, **22**, **23**, **24**, **50** und **70** heruntergeladen und/oder gespeichert wird.

[0027] Jeder der Regler **12a** und **16a**, der beispielsweise der Regler Delta V™ sein kann, der von Emerson Process Management vermarktet wird, kann eine Regleranwendung speichern und ausführen, die unter Verwendung einer gewissen Anzahl von unterschiedlichen, unabhängig ausgeführten Regelmodulen oder Blöcken eine Regelstrategie umsetzt. Die Regelmodule können jeweils aus dem bestehen, was man gewöhnlich als Funktionsblöcke bezeichnet, wobei jeder Funktionsblock ein Teil oder eine Nebenroutine einer globalen Regelungsroutine ist und in Zusammenhang mit anderen Funktionsblöcken (über Kommunikationsmittel, die als Verbindungen bezeichnet werden) funktioniert, um Prozessregelschleifen innerhalb der Prozessanlage **10** umzusetzen. Wie es wohlbekannt ist, führen die Funktionsblöcke typischerweise eine Eingabefunktion (wie etwa diejenige, die mit einem Transmitter, einem Sensor oder einer anderen Prozessparameter-Messvorrichtung verknüpft ist), eine Regelfunktion (wie etwa diejenige, die mit einer Regelungsroutine verknüpft ist, die diverse Regelungen ausführt, wie etwa PID, unscharfe Logik usw.) oder eine Ausgabefunktion, die den Betrieb eines Geräts (wie etwa eines Ventils) regelt, um eine gewisse physische Funktion innerhalb der Prozessanlage **10** auszuführen, aus. Natürlich gibt es hybride und andere Arten von Funktionsblöcken, die verwendet werden können. Obwohl ein Fieldbus-Protokoll und das Systemprotokoll Delta V™ Regelmodule und Funktionsblöcke verwenden können, die in einem objektorientierten Programmierprotokoll ausgelegt und umgesetzt sind, könnten die Regelmodule unter Verwendung einer beliebigen gewünschten Regelungsprogrammiermethode ausgelegt werden, wozu beispielsweise sequenzielle Funktionsblöcke, Strichleiterlogik usw. gehören, und sind nicht darauf eingeschränkt, unter Verwendung von Funktionsblöcken oder einer beliebigen anderen bestimmten Programmiertechnik ausgelegt zu werden. Typischerweise kann die Konfiguration der Regelmodule, wie sie in den Prozessregelknoten **12** und **16** gespeichert ist, in der Konfigurationsdatenbank **32**

gespeichert werden, die für Anwendungen zugänglich ist, die von den Arbeitsstationen **18a** und **20a** ausgeführt werden. Die Funktionsblöcke können beispielsweise in dem Regler **12a**, **16a** gespeichert sein und ausgeführt werden, wie es typischerweise der Fall ist, wenn diese Funktionsblöcke für standardmäßige 4-20mA-Geräte und gewisse Arten von intelligenten Feldgeräten, wie etwa HART-Geräten, verwendet werden oder damit verknüpft sind, oder können in den Feldgeräten selber gespeichert sein und umgesetzt werden, wie es der Fall bei Fieldbus-Geräten sein kann.

[0028] Bei dem in **Fig.** 1 abgebildeten System können die Feldgeräte **22** und **23**, die mit den Reglern **12a** und **16a** gekoppelt sind, standardmäßige 4-20 mA-Geräte sein, oder können intelligente Feldgeräte sein, wie etwa HART-, Profibus- oder Foundation Fieldbus-Feldgeräte, die einen Prozessor und einen Speicher umfassen. Einige dieser Geräte, wie etwa Foundation Fieldbus-Feldgeräte (in **Fig.** 1 mit der Bezugszahl **23** bezeichnet), können Module oder Nebenmodule, wie etwa Funktionsblöcke, die mit der Regelstrategie verknüpft sind, die in den Reglern **12a** und **16a** umgesetzt wird, speichern und ausführen. Natürlich können die Feldgeräte **22**, **23** beliebige Arten von Geräten sein, wie etwa Sensoren, Ventile, Transmitter, Positionierer usw., und die E/A-Geräte **24** können beliebige Arten von E/A-Geräten sein, die sich einem gewünschten Kommunikations- oder Reglerprotokoll anpassen, wie etwa HART, Foundation Fieldbus, Profibus usw.

[0029] Die Regler **12a** und **16a** umfassen jeweils einen Prozessor, der eine oder mehrere Prozessregelungsroutinen umsetzt oder beaufsichtigt, die in einem Speicher gespeichert sind und Regelschleifen umfassen können, die darin gespeichert sind oder anderweitig damit verknüpft sind. Die Regler **12a** und **16a** kommunizieren mit den Feldgeräten **22**, **23**, den Arbeitsstationen **18a**, **20a** und der Datenbank **32**, um einen Prozess auf eine gewünschte Art und Weise zu regeln. Die Regler **12a** und **16a** können jeweils konfiguriert sein, um eine Regelstrategie oder eine Regelungsroutine auf beliebige Art und Weise umzusetzen. Es versteht sich, dass die Regler **12a** und **16a** mit den Feldgeräten **22**, **23**, den Arbeitsstationen **18a**, **20a** und der Datenbank **32** über eine drahtlose Verbindung in Verbindung stehen können.

[0030] Die Prozessanlage **10** kann auch ein Sicherheitssystem **14** (mit Punktlinien angegeben) umfassen, das mit den Prozessregelknoten **12** und **16** integriert ist. Das Sicherheitssystem **14** kann im Allgemeinen als sicherheitstechnisches System (SIS) funktionieren, um die Regelung zu überwachen oder zu übergehen, die von den Prozessregelknoten **12** und **16** bereitgestellt wird, um den wahrscheinlichen sicheren Betrieb der Prozessanlage **10** zu maximieren.

[0031] Jeder der Knoten **12** und **16** kann einen oder mehrere Logic Solver **50** des Sicherheitssystems umfassen. Jeder der Logic Solver **50** ist ein E/A-Gerät, das über einen Prozessor und einen Speicher verfügt, und ist konfiguriert, um logische Sicherheitsmodule auszuführen, die in dem Speicher abgelegt sind. Jeder Logic Solver **50** ist kommunikationsmäßig gekoppelt, um den Feldgeräten **60** und **62** des Sicherheitssystems Regelsignale bereitzustellen und/oder um Signale von ihnen zu empfangen. Zusätzlich umfasst jeder der Knoten **12** und **16** mindestens ein Nachrichtenverbreitungsgerät (MPD) **70**, das über einen Ring- oder Busanschluss **74** (von dem in **Fig.** 1 nur ein Teil abgebildet ist) kommunikationsmäßig mit anderen MPD **70** verbunden ist. Der Logic Solver **50** des Sicherheitssystems, die Feldgeräte **60** und **62** des Sicherheitssystems, die MPD **70** und der Bus **74** bilden allgemein das Sicherheitssystem **14** aus **Fig.** 1.

[0032] Die Logic Solver **50** aus **Fig.** 1 können eine beliebige gewünschte Art von Sicherheitssystem-Regelgeräten sein, wozu ein Prozessor und ein Speicher gehören, der logische Sicherheitsmodule speichert, die geeignet sind, um auf dem Prozessor ausgeführt zu werden, um eine Regelfunktionalität, die mit dem Sicherheitssystem **14** verknüpft ist, unter Verwendung der Feldgeräte **60** und **62** bereitzustellen. Natürlich können die Sicherheitsfeldgeräte **60** und **62** eine beliebige gewünschte Art von Feldgeräten sein, die einem beliebigen bekannten oder gewünschten Kommunikationsprotokoll entsprechen oder dieses verwenden, wie etwa die zuvor erwähnten. Insbesondere können die Feldgeräte **60** und **62** sicherheitstechnische Feldgeräte von der Art sein, die herkömmlicherweise von einem separaten, dedizierten, sicherheitstechnischen Regelungssystem geregelt wird. In der Prozessanlage **10**, die in **Fig.** 1 abgebildet ist, sind die Sicherheitsfeldgeräte **60** abgebildet, wie sie ein dediziertes oder Punkt-zu-Punkt-Kommunikationsprotokoll, wie etwa das HART- oder das 4-20mA-Protokoll, verwenden, während die Sicherheitsfeldgeräte **62** abgebildet sind, wie sie ein Buskommunikationsprotokoll, wie etwa ein Fieldbus-Protokoll, verwenden. Die Sicherheitsfeldgeräte **60** können eine beliebige gewünschte Funktion ausführen, wie etwa die eines Sperrventils, eines Ausschalters usw.

[0033] Eine gemeinsame Rückwandplatine (nicht gezeigt) kann in jedem der Knoten **12** und **16** verwendet werden, um die Regler **12a** und **16a** mit den E/A-Karten **24** der Prozessregelung, den Sicherheits-Logic-Solvern **50** und den MPD **70** kommunikationsmäßig zu koppeln. Die Regler **12a** und **16a** sind ebenfalls kommunikationsmäßig mit dem Netzwerk **30** gekoppelt. Die Regler **12a** und **16a**, die E/A-Geräte **24**, die Logic Solver **50** und die MPD **70** können mit den Knoten **18** und **20** über das Netzwerk **30** kommunizieren.

[0034] Wie es der Fachmann verstehen wird, ermöglicht es die (nicht gezeigte) Rückwandplatine in dem Knoten **12**, **16** den Logic Solvern **50**, lokal miteinander zu kommunizieren, um die Sicherheitsfunktionen zu koordinieren, die von diesen Geräten umgesetzt werden, um einander Daten zu übermitteln und/oder um andere integrierte Funktionen auszuführen. Ähnlich ermöglicht es die (nicht gezeigte) Rückwandplatine in dem Knoten **16** den Logic Solvern **50**, lokal miteinander zu kommunizieren, um die Sicherheitsfunktionen zu koordinieren, die von diesen Geräten umgesetzt werden, um einander Daten zu übermitteln und/oder um andere integrierte Funktionen auszuführen. Andererseits funktionieren die MPD **70**, um es Teilen des Sicherheitssystems **14** zu ermöglichen, die in völlig verschiedenen Positionen der Anlage **10** angeordnet sind, weiterhin miteinander zu kommunizieren, um einen koordinierten Sicherheitsbetrieb an verschiedenen Knoten der Prozessanlage **10** bereitzustellen. Insbesondere ermöglichen es die MPD **70** zusammen mit dem Bus **74** den Logic Solvern **50**, die mit verschiedenen Knoten **12** und **16** der Prozessanlage **10** verknüpft sind, kommunikationsmäßig zusammen kaskadiert zu sein, um die Kaskadierung von sicherheitstechnischen Funktionen innerhalb der Prozessanlage **10** gemäß einer zugewiesenen Priorität zu ermöglichen. Die MPD **70** und der Bus **74** versorgen das Sicherheitssystem mit einer Kommunikationsverbindung, die eine Alternative zu dem Netzwerk **30** ist.

[0035] Alternativ können zwei oder mehrere sicherheitstechnische Funktionen an unterschiedlichen Positionen innerhalb der Prozessanlage **10** miteinander zusammenhängen oder zusammengeschaltet sein, ohne dass eine dedizierte Leitung zu einzelnen Sicherheitsfeldgeräten in den getrennten Bereichen oder Knoten der Anlage **10** zu legen wäre. Mit anderen Worten ermöglicht es die Verwendung der MPD **70** und **72** und des Busses **74** einem Sicherheitstechniker, ein Sicherheitssystem **14** auszulegen und zu konfigurieren, das über die gesamte Prozessanlage **10** verteilt ist, doch dessen verschiedene Komponenten kommunikationsmäßig zusammengeschaltet sind, damit die ungleichartige sicherheitstechnische Hardware je nach Bedarf miteinander kommunizieren kann. Dieses Merkmal stellt auch eine Skalierbarkeit des Sicherheitssystems **14** bereit, indem es die Möglichkeit bietet, zusätzliche Sicherheits-Logic Solver zu dem Sicherheitssystem **14** hinzuzufügen, je nachdem wie sie benötigt werden, oder wenn neue Prozessregelknoten zu der Prozessanlage **10** hinzugefügt werden.

[0036] **Fig.** 2 ist ein Blockdiagramm einer beispielhaften Arbeitsstation **18a** (die Arbeitsstation **20a** kann die gleiche oder eine ähnliche Vorrichtung umfassen). Die Arbeitsstation **18a** kann mindestens einen Prozessor **100**, einen flüchtigen Speicher **104** und einen nicht flüchtigen Speicher **108** umfassen. Der flüchtige Speicher **104** kann beispielsweise einen Arbeitsspeicher (RAM) umfassen. Bei einigen Ausführungsformen kann der RAM mit einer oder mehreren Batterien gesichert werden, so dass die Daten bei einem Stromausfall nicht verloren gehen. Der nicht flüchtige Speicher **108** kann beispielsweise eines oder mehreres von einer Festplatte, einem Festspeicher (ROM), einer CD-ROM, einem programmierbaren ROM (PROM), einem löschbaren programmierbaren ROM (EPROM), einem elektrisch löschbaren programmierbaren ROM (EEPROM), einer DVD („Digital Versatile Disk"), einem Flash-Speicher usw. umfassen. Die Arbeitsstation **18a** kann auch ein Arbeitsstation-E/A-Gerät **112** umfassen. Der Prozessor **100**, der flüchtige Speicher **104**, der nicht flüchtige Speicher **108** und das Arbeitsstation E/A-Gerät **112** können über einen Adressen-/Datenbus **116** zusammengeschaltet sein. Die Arbeitsstation **18a** kann auch mindestens ein Anzeigegerät **120** und mindestens ein Benutzereingabegerät **124** umfassen, das beispielsweise ein oder mehreres von einer Tastatur, einem Tastenfeld, einer Maus, einer Steuerkugel, einem Berührungsbildschirm, einem Lichtgriffel usw. sein kann. Bei einigen Ausführungsformen kann bzw. können einer oder mehrere von dem flüchtigen Speicher **104**, dem nicht flüchtigen Speicher **108** und der Arbeitsstation-E/A-Gerät **112** mit dem Prozessor **100** über einen Bus gekoppelt sein, der von dem Adressen-/Datenbus **116** getrennt ist (nicht gezeigt), oder kann bzw. können direkt mit dem Prozessor **100** gekoppelt sein.

[0037] Das Anzeigegerät **120** und das Benutzereingabegerät **124** sind mit dem Arbeitsstation-E/A-Gerät **112** gekoppelt. Zusätzlich ist die Arbeitsstation **18a** mit dem Netzwerk **30** über das Arbeitsstation-E/A-Gerät **112** gekoppelt. Obwohl das Arbeitsstation-E/A-Gerät **112** in **Fig.** 2 als ein einziges Gerät abgebildet ist, kann es mehrere Geräte umfassen. Zusätzlich kann bzw. können bei einigen Ausführungsformen ein oder mehrere des Anzeigegeräts **120** und des Benutzereingabegeräts **124** direkt mit dem Adressen-/Datenbus **116** oder dem Prozessor **100** gekoppelt sein.

[0038] Nun mit Bezug auf **Fig.** 1 und **Fig.** 2 kann eine Anwendung zur Konfiguration der Prozessregelung, die mit einem oder mehreren der Regelknoten **12**, **16** gekoppelt ist, auf einer oder mehreren der Arbeitsstationen **18a** und **20a** gespeichert sein und von dieser bzw. diesen ausgeführt werden. Beispielsweise könnte die Anwendung zur Konfiguration der Prozessregelung in dem nicht flüchtigen Speicher **108** und/oder dem flüchtigen Speicher **104** gespeichert sein und von dem Prozessor **100** ausgeführt werden. Soweit erwünscht, könnte diese Anwendung jedoch in anderen Computer gespeichert sein und ausgeführt werden, die mit der Prozessanlage **10** verknüpft sind. Im Allgemeinen ermöglicht es die Anwendung zur Konfiguration der Prozessregelung ei-

nem Programmierer, Regelungsroutinen, Regelmodule, Funktionsblöcke, Programme, Logik usw. zu erstellen und zu konfigurieren, die von den Reglern **12a**, **16a**, den E/A-Geräten **24** und/oder den Feldgeräten **22**, **23** umzusetzen sind. Diese Regelungsroutinen, Regelmodule, Funktionsblöcke, Programme, Logik usw. können dann in ein entsprechendes Exemplar von den Reglern **12a**, **16a**, den E/A-Geräten **24** und/oder den Feldgeräten **22**, **23** über das Netzwerk **30** heruntergeladen werden.

[0039] Ähnlich kann eine Anwendung zur Konfiguration des Sicherheitssystems, die mit dem Sicherheitssystem **14** verknüpft ist, auf einer oder mehreren der Arbeitsstationen **18a** und **20a** gespeichert sein und ausgeführt werden. Beispielsweise könnte die Anwendung zur Konfiguration des Sicherheitssystems in dem nicht flüchtigen Speicher **108** und/oder dem flüchtigen Speicher **104** gespeichert sein und von dem Prozessor **100** ausgeführt werden. Soweit erwünscht könnte diese Anwendung jedoch in anderen Computer gespeichert sein und ausgeführt werden, die mit der Prozessanlage **10** verknüpft sind. Im Allgemeinen ermöglicht es die Anwendung zur Konfiguration des Sicherheitssystems einem Programmierer, Regelungsroutinen, Regelmodule, Funktionsblöcke, Programme, Logik usw. zu erstellen und zu konfigurieren, die von den Reglern **12a**, **16a**, den Logic Solvern **50** und/oder den Geräten **60**, **62** umzusetzen sind. Diese Regelungsroutinen, Regelmodule, Funktionsblöcke, Programme, Logik usw. können dann auf geeignete Exemplare von den Reglern **12a**, **16a**, den Logic Solvern **50** und/oder den Geräten **60**, **62** über das Netzwerk **30** heruntergeladen werden.

Zustandsmaschinen-Funktionsblock

[0040] Eine Anwendung zur Konfiguration eines Regelungssystems oder Sicherheitssystems kann das Programmieren von Regelmodulen und/oder Regelungsroutinen unter Verwendung eines Funktionsblock-Programmierparadigmas ermöglichen. **Fig.** 3 bildet ein Beispiel einer Anzeige **150** ab, die ein Regelmodul **154** darstellt. Die Anzeige **150** kann Teil einer Benutzerschnittstelle sein, die mit der Konfigurationsanwendung verknüpft ist, und die Anzeige **150** kann einem Programmierer beispielsweise über das Anzeigegerät **120** der Arbeitsstation **18a** präsentiert werden. Die Anzeige **150** stellt das Regelmodul **154** dar, das einen Satz kommunikationsmäßig zusammengeschalteter Funktionsblöcke aufweist, die auf entsprechenden Exemplaren von den Reglern **12a**, **16a**, den E/A-Geräten **24**, den Logic Solvern **50** und/oder den Geräten **22**, **23**, **60**, **62** erstellt und zum Umsetzen während der Funktionsweise einer Prozessanlage über das Netzwerk **30** darauf heruntergeladen werden können. Wie in **Fig.** 3 abgebildet, umfasst das Regelmodul **154** einen Zustandsmaschinen-Funktionsblock (SMFB) **160**, eine Vielzahl von analogen Eingabe-(AI) und digitalen Eingabe-(DI)Funktionsblöcken, eine Vielzahl von analogen Ausgabe-(AO) und digitalen Ausgabe-(DO)Funktionsblöcken und anderen Funktionsblöcken (FB). Der SMFB **160** verfügt über Eingänge, die kommunikationsmäßig mit den Funktionsblöcken **114** zusammengeschaltet sind, die beispielsweise DI-Funktionsblöcke oder andere FB sein können. Der SMFB **160** verfügt auch über Ausgänge, die an die Funktionsblöcke **118** angeschlossen sind, die beispielsweise DO-Funktionsblöcke oder andere FB sein können. Das Regelmodul **154** kann eines von einer Vielzahl von Regelmodulen, die zusammen Geräte, wie etwa Schalter, Ventile usw. regeln, als Teil eines Regelungssystems, eines Sicherheitssystems usw. regeln oder eines davon sein. Natürlich ist das Regelmodul **154** nur ein Beispiel eines Regelmoduls, das SMFB verwendet. Im Allgemeinen kann ein Regelmodul auf beliebige gewünschte Art und Weise programmiert sein, um beliebige Arten von Funktionsblöcken zu umfassen, die kommunikationsmäßig mit einer beliebigen Anzahl von SMFB auf beliebige Art und Weise gekoppelt sind, und auf beliebige gewünschte oder nützliche Art und Weise konfiguriert sind, um eine beliebige gewünschte Funktion auszuführen. Falls es beispielsweise in einem Fieldbus-Netzwerk verwendet wird, kann ein Regelmodul eine beliebige Art von Fieldbus-Funktionsblöcken umfassen.

[0041] Bei einigen Ausführungsformen kann bzw. können ein oder mehrere der Eingaben für den SMFB **160** anders als von einem Funktionsblock empfangen werden. Beispielsweise kann bzw. können mehrere der Eingänge zu dem SMFB **160** kommunikationsmäßig gekoppelt sein, um Eingaben von einem Bediener beispielsweise über eine Bedienerschnittstelle zu empfangen. Beispielsweise könnte ein Bediener unter Verwendung einer Bedienerschnittstelle, die an einem Knoten umgesetzt wird, wie etwa dem Knoten **18** oder **20**, Eingaben für den SMFB **160** bereitstellen.

[0042] Der SMFB kann ein Funktionsblock sein, der eine Zustandsmaschine umsetzt. Bei einigen Ausführungsformen kann eine Zustandsmaschine eine Entität (z. B. ein Gerät, eine Software, die von einem Prozessor umgesetzt wird, usw.) umfassen, die sich in einem von einer Vielzahl von Zuständen befinden kann. Die Zustandsmaschine kann von einem Zustand in einen andern Zustand übergehen, falls eine bestimmte Eingabe für die Zustandsmaschine erfolgt. Der SMFB kann Ausgaben bereitstellen, die auf dem aktuellen Zustand der Zustandsmaschine basieren. Als nur ein Beispiel kann der SMFB eine oder mehrere Ausgaben bereitstellen, die den aktuellen Zustand der Zustandsmaschine angibt bzw. angeben. Noch allgemeiner kann eine Zustandsmaschine eine Entität umfassen (z. B. ein Gerät, eine Software, die von einem Prozessor umgesetzt wird, usw.), die einen Status der Entität oder einer gewissen anderen Entität (z. B. eine Prozessanlage, einen Nebenteil einer Prozessanlage, eine Komponente einer Prozessanlage usw.)

zu einem bestimmten Zeitpunkt speichert und die den Status ändern kann und/oder bewirken kann, dass eine Aktion oder eine Ausgabe basierend auf den Eingaben in die Zustandsmaschine erfolgt.

[0043] Unter Verwendung der Benutzerschnittstelle, die mit der Konfigurationsanwendung verknüpft ist, kann der Programmierer ein Regelmodul, wie etwa das Regelmodul **154**, auslegen. Als nur ein Beispiel kann die Benutzerschnittstelle einen Mechanismus bereitstellen, damit ein Programmierer gewünschte Funktionsblöcke beispielsweise aus einer Schablone oder einer Palette, die eine Vielzahl von standardmäßigen oder spezifisch angepassten Funktionsblockschablonen umfasst, auswählt. Zusätzlich kann die Benutzerschnittstelle ein grafisches Diagramm bereitstellen, auf dem der Programmierer Abbildungen von Funktionsblöcken einfügen oder einsetzen kann. Der Programmierer kann beispielweise eine Maus, eine Steuerkugel, eine Tastatur, ein Tastenfeld, einen Berührungsbildschirm usw. verwenden, um einen Funktionsblock aus der Schablone oder der Palette auszuwählen und dann den Funktionsblock auf das grafische Diagramm zu „ziehen und abzulegen". Der Programmierer kann zusätzlich Funktionsblöcke kommunikationsmäßig koppeln, indem er beispielsweise unter Verwendung beispielsweise einer Maus, einer Steuerkugel, einer Tastatur, eines Tastenfeldes, eines Berührungsbildschirms usw. eine Linie zwischen einem Ausgang eines Funktionsblocks und einem Eingang eines anderen Funktionsblocks zieht.

[0044] Sobald es konfiguriert ist, kann das Regelmodul **154** beispielsweise durch einen oder mehrere der Regler **12a**, **14a**, **16a**, der E/A-Geräte **24**, der Logic Solver **50** und der Geräte **22**, **23**, **60**, **62** umgesetzt werden.

[0045] **Fig.** 4 ist ein Beispiel einer Darstellung eines SMFB **200**, der beispielsweise an einer Benutzerschnittstellenanzeige, wie etwa der Anzeige **150** aus **Fig.** 3, angezeigt werden kann. Die Darstellung des SMFB **200** gibt an, dass der SMFB **200** sieben Dateneingaben (IN_D1 bis IN_D7) und sieben Datenausgaben (TRANS_OUT, TRANS_IN, STATE und OUT_D1 bis OUT_D6) umfasst. Die Dateneingaben können im Allgemeinen Bedingungen innerhalb der Prozessanlage angeben, können Bedienerbefehle usw. angeben und können bewirken, dass sich die Zustände einer Zustandsmaschine, die von dem SMFB **200** umgesetzt wird, ändern. Die Datenausgaben können einen oder mehrere Indikatoren des Zustandes der Zustandsmaschine, die dem SMFB **200** entsprechen, sowie Konfigurationselemente, die Funktionen oder Aktionen entsprechen, die basierend auf dem Zustand auszuführen sind, umfassen. Beispielsweise kann die Ausgabe STATE ein Indikator des Zustands (z. B. Zustand 1, Zustand 2, Zustand 3 usw.) der Zustandsmaschine sein. Die Ausgabe OUT_D1 kann ein Indikator dafür sein, ob sich die Zustandsmaschine in einem Zustand „Zustand 1" befindet. Ähnlich können die Ausgaben OUT_D2, OUT_D3, ... OUT_D6 Indikatoren dafür sein, ob sich die Zustandsmaschine jeweils in den Zuständen „Zustand 2", „Zustand 3" ... „Zustand 6" befindet. Zusätzlich kann die Ausgabe TRANS_OUT ein Konfigurationselement angeben, das auszuführen ist, wenn die Zustandsmaschine einen Übergang aus einem bestimmten Zustand vornimmt, und die Ausgabe TRANS_IN kann ein Konfigurationselement angeben, das auszuführen ist, wenn die Zustandsmaschine einen Übergang in einen bestimmten Zustand vornimmt. Bei einigen Ausführungsformen kann der SMFB **200** eine Vielzahl von Ausgaben TRANS_OUT und eine Vielzahl von Ausgaben TRANS_IN umfassen. Beispielsweise kann der SMFB **200** eine Anzahl von Ausgaben TRANS_OUT und eine Anzahl von Ausgaben TRANS_IN umfassen, die gleich der Anzahl von Zuständen sind (d. h. eine Ausgabe TRANS_OUT und eine Ausgabe TRANS_IN pro Zustand). Es versteht sich, dass der SMFB **200** eine beliebige Anzahl von Ausgaben TRANS_OUT und TRANS_IN umfassen kann.

[0046] Der SMFB **200** kann auch andere Eingaben neben Dateneingaben umfassen, wie etwa eine Eingabe ENABLE, eine Eingabe TRK_VAL und eine Eingabe TRK_IN_D. Beispielsweise kann der SMFB **200** eine Eingabe von einem anderen SMFB aufweisen. Ferner kann der SMFB **200** auch andere Ausgaben neben den Ausgaben umfassen, die den Zustand oder damit verknüpfte Konfigurationselemente angeben. Die Eingaben ENABLE, TRK_VAL und TRK_IN_D werden nachstehend ausführlicher beschrieben. Obwohl der SMFB **200** in **Fig.** 4 gezeigt wird, wie er sieben Dateneingaben und neun Datenausgaben aufweist, können andere Ausführungsformen eine beliebige gewünschte Anzahl von Dateneingaben und Datenausgaben umfassen. Die Anzahl von Dateneingaben und die Anzahl von Datenausgaben des SMFB **200** können konfigurierbar sein oder nicht. Bei einer Ausführungsform entspricht die Anzahl der Ausgaben OUT_Dx im Allgemeinen der Anzahl von möglichen Zuständen der Zustandsmaschine, die von dem SMFB **200** umgesetzt wird, und die Anzahl von möglichen Zuständen kann konfigurierbar sein. Die Anzahl von Ausgaben OUT_D1, OUT_D2 usw. muss jedoch nicht unbedingt der Anzahl von möglichen Zuständen der Zustandsmaschine entsprechen. Falls es beispielweise weniger Zustände als die Anzahl von Ausgaben OUT_D1, OUT_D2 usw. gibt, können die übrigen Ausgaben unbenutzt bleiben.

[0047] Unter Verwendung der Benutzerschnittstelle, die mit dem Konfigurationsprogramm verknüpft ist, kann der Programmierer einen oder mehrere Funktionsblöcke, wie etwa den SMFB **200**, konfigurieren. Mit Bezug auf das Konfigurieren des SMFB kann der Programmierer eine Anzahl von möglichen Zuständen vorgeben, wie die Eingaben bewirken, dass

die Zustandsmaschine zwischen Zuständen übergeht, und beliebige Funktionen oder Aktionen, die bevor, während oder nachdem die Zustandsmaschine zwischen den Zuständen übergeht bzw. übergegangen ist, auszuführen sind. Um es einem Programmierer zu erlauben, den SMFB **200** zu konfigurieren, kann eine Konfigurationsanwendung auf dem Anzeigegerät **120** einen Benutzerschnittstellen-Mechanismus, wie etwa ein Konfigurationsfenster, einen Bildschirm usw., anzeigen, der mit dem Funktionsblock verknüpft ist.

[0048] **Fig.** 5 ist ein Beispiel eines Benutzerschnittstellen-Mechanismus, der verwendet werden kann, um einen SMFB, wie etwa den SMFB **200** aus **Fig.** 4, mindestens teilweise zu konfigurieren. Der Benutzerschnittstellen-Mechanismus umfasst eine Tabelle oder Matrix **300** (nachstehend einfach als „Matrix **300**" bezeichnet), die als Teil eines Konfigurationsfensters, eines Bildschirms usw., das bzw. der mit dem SMFB verknüpft ist, angezeigt werden kann. Die Matrix **300** umfasst eine Vielzahl von Zellen (**302**, **303**, **304**), die in Reihen und Spalten angeordnet sind. Wie gezeigt, sind die Spalten in Dreiergruppen angeordnet, wobei jede Gruppe einem von einer Vielzahl möglicher Zustände einer Zustandsmaschine entsprechen kann. Insbesondere entspricht die mittlere Spalte innerhalb jeder Gruppe einem bestimmten Zustand der Zustandsmaschine, wie er in herkömmlichen Zustandsdiagrammen enthalten ist. Beispielsweise wie in **Fig.** 5 gezeigt, ist der mit „1" bezeichnete Zustand der Zustand „AUSGELÖST", der mit „2" bezeichnete Zustand ist der Zustand „WARTEN AUF ZURÜCKSETZEN" und so weiter. Gemäß den Ausführungsformen entspricht die linke Spalte in jeder Gruppe einem Konfigurationselement, das die Zustandsmaschine ausführen soll, wenn sie in einen nächsten Zustand übergeht („Übergang-in-Aktion"), und die rechte Spalte in jeder Gruppe entspricht einem Konfigurationselement, das die Zustandsmaschine ausführen soll, wenn sie aus einem derzeitigen Zustand übergeht („Übergang-aus-Aktion"). Ferner entspricht jede Reihe einer Eingabe in die Zustandsmaschine (z. B. „ANFANG", "ZURÜCKSETZEN ERLAUBNIS" usw.). Entsprechend gibt jede der Zellen **302** (die sich für jede Reihe und Spaltengruppe wiederholen) eine Übergang-in-Aktion vor, jede der Zellen **303** (die sich für jede Reihe und Spaltengruppe wiederholen) gibt ein Eingabe/Zustands-Paar an, und jede der Zellen **304** (die sich für jede Reihe und Spaltengruppe wiederholen) gibt eine Übergang-aus-Aktion an. Es versteht sich, dass eine Gruppe von Zellen **302**, **303**, **304** zu einer einzigen Zelle zusammengefasst werden kann. Ferner versteht es sich, dass jede der Zellen **302**, **303**, **304** eine Teilzelle für eine Zelle sein kann, die einem bestimmten Zustand entspricht. Obwohl die beispielhafte Matrix **300** Reihen für sieben Eingaben und sechs Zustandsgruppen umfasst, können ähnliche Matrizen, die andere Anzahlen von Gruppen von Zuständen und Ausgaben aufweisen, für SMFB verwendet werden, die andere Anzahlen von Eingaben und Gruppen von Zuständen aufweisen. Die Anzahl von Eingaben und Gruppen von Zuständen kann konfigurierbar sein. Bei anderen Beispielen können die Reihen einem von einer Vielzahl von möglichen Zuständen der Zustandsmaschine entsprechen (sowie den damit verknüpften Übergang-in- und Übergang-aus-Aktionen), und jede Spalte kann einer Eingabe für die Zustandsmaschine entsprechen.

[0049] Im Betrieb kann die Zustandsmaschine in Abhängigkeit von einer Eingabe, die an der Zustandsmaschine bestätigt wird, wenn sie sich in dem aktuellen Zustand befindet, von einem aktuellen Zustand in einen nächsten Zustand übergehen. Beispielsweise kann die Matrix **300** vorgeben, dass wenn sich die Zustandsmaschine in dem Zustand „NORMALER BETRIEB" befindet (aktueller Zustand) und die Eingabe „AUSLÖSUNG ANGEFRAGT" bestätigt ist, die Zustandsmaschine in den Zustand „AUSGELÖST" (nächsten Zustand) übergehen soll. In manchen Fällen können gewisse Zustände eventuell keinen nächsten Zustandsübergang für eine gewisse Eingabe vorgeben.

[0050] Gemäß den Ausführungsformen kann die Übergang-aus-Aktion, die von der rechten Spalte in jeder Gruppe vorgegeben wird, einem oder mehreren Konfigurationselementen entsprechen, das bzw. die der SMFB gemäß dem Übergang aus einem aktuellen Zustand ausführt, und die Übergang-in-Aktion, die von der linken Spalte in der Gruppe vorgegeben wird, kann einem oder mehreren Konfigurationselementen entsprechen, das bzw. die der SMFB gemäß dem Übergang in einen nächsten Zustand ausführt. Bei einigen Ausführungsformen kann das eine Konfigurationselement oder können die mehreren Konfigurationselemente eine Logik in der Form von strukturiertem Text oder einer übergeordneten Computerkonstruktion (z. B. C, C++, JAVA usw.), der bzw. die auszuführende Aktionen definiert, vorliegen. Es versteht sich, dass das eine oder die mehreren Konfigurationselemente in Form einer beliebigen Art von Code oder ausführbarer Logik vorliegen kann. Das eine oder die mehreren Konfigurationselemente kann bzw. können ein einziges „Einmal-"Element sein, das der SMFB bei einem Übergang von einem aktuellen Zustand in einen nächsten Zustand ausführt, wodurch in manchen Fällen das eine oder die mehreren Konfigurationselemente nicht weiter funktioniert bzw. funktionieren, sobald der nächste Zustand erreicht ist. Der SMFB kann zuerst die Übergang-aus-Aktion (d. h. das Verlassen des aktuellen Zustands bewirkt, dass der SMFB die Übergang-aus-Aktion ausführt) unabhängig davon ausführen, wohin der Übergang des SMFB geht, und der SMFB kann, bevor er sich in dem nächsten Zustand stabilisiert, unabhängig davon, woher der Übergang des SMFB kommt, die Übergang-in-Aktion ausführen (d.

h. das Eintreten in den nächsten Zustand bewirkt, dass der SMFB die Übergang-in-Aktion ausführt). In manchen Fällen können die Konfigurationselemente (d. h. die Übergang-in- und Übergang-aus-Aktionen) auf andere Komponenten oder Entitäten außerhalb des SMFB einwirken (z. B. auf einen anderen Funktionsblock). Es versteht sich, dass andere zeitliche Komponenten, die mit den Übergang-in-Aktionen, den Übergang-aus-Aktionen und Zustandsänderungen verknüpft sind, in Betracht gezogen werden.

[0051] Mit Bezug auf **Fig.** 4 entsprechen die Eingaben „1" bis „7" der Matrix **300** jeweils den Eingaben IN_D1 bis IN_D7 des SMFB **200**. Ähnlich entsprechen die Zustände „1" bis „6" der Matrix **300** jeweils den Ausgaben OUT_D1 bis OUT_D6 des SMFB **200**. Zusätzlich kann bei diesem Beispiel ein Programmierer in der Lage sein, jeden möglichen Zustand und/oder jede der Eingaben zu bezeichnen. Beispielsweise ist in **Fig.** 5 der „Zustand 1" mit „AUSGELÖST" bezeichnet, und die Eingabe 1 ist mit „ANFANG" bezeichnet. Das Bezeichnen der Eingaben und/oder Zustände kann dazu beitragen, das Verständnis der Funktionsweise der Zustandsmaschine zu erleichtern.

[0052] Ein Programmierer kann den SMFB **200** konfigurieren, indem er Konfigurationsinformationen in die Zellen **302**, **303**, **304** einträgt. Insbesondere für eine bestimmte Zelle, die einem Eingabe/Zustands-Paar entspricht, kann der Programmierer Konfigurationsdaten in die bestimmte Zelle eintragen, die den Zustand angeben, in den der SMFB **200** übergehen soll. Ferner kann der Programmierer für eine bestimmte Zelle, die einer Übergang-in-Aktion entspricht, Konfigurationsdaten in die bestimmte Zelle eintragen, die eine Aktion angeben, die der SMFB **200** ausführen soll, wenn er in einen nächsten Zustand übergeht. Ferner kann der Programmierer noch für eine bestimmte Zelle, die einer Übergang-aus-Aktion entspricht, Konfigurationsdaten in die bestimmte Zelle eintragen, die eine Aktion angeben, die der SMFB **200** ausführen soll, wenn er aus einem aktuellen Zustand übergeht.

[0053] Der SMFB **200** kann die Ausgabe TRANS_OUT basierend auf entsprechenden Übergang-aus-Konfigurationsdaten bestätigen, um eine Funktion auszuführen, nachdem die Zustandsmaschine aus einem aktuellen Zustand übergegangen ist. Bei einigen Ausführungsformen kann der SMFB **200** die Ausgabe TRANS_OUT bestätigen, bevor die Zustandsmaschine aus dem aktuellen Zustand übergeht. Der SMFB **200** kann die Ausgabe TRANS_IN basierend auf entsprechenden Übergang-in-Konfigurationsdaten bestätigen, nachdem die Zustandsmaschine aus einem aktuellen Zustand übergegangen ist (oder ansonsten bevor sie aus dem aktuellen Zustand übergeht) und bevor sich die Zustandsmaschine in dem nächsten Zustand stabilisiert. Bei einigen Ausführungsformen kann der SMFB **200** die Ausgabe TRANS_IN bestätigen, nachdem sich die Zustandsmaschine in dem nächsten Zustand stabilisiert hat. In manchen Fällen kann der SMFB **200** ein Konfigurationselement bereitstellen, das einer Übergang-aus-Aktion oder einer Übergang-in-Aktion als Eingabe in einen zusätzlichen SMFB oder ein Prozessregelgerät entspricht, um zu bewirken, dass der zusätzliche SMFB oder das zusätzliche Prozessregelgerät eine Funktion erfüllt, während der SMFB **200** entweder aus dem aktuellen Zustand übergeht oder in den nächsten Zustand übergeht. Entsprechend kann eine Aktivierung einer Ausgabe TRANS_OUT eines ersten SMFB bewirken, dass ein zweiter SMFB in einen nächsten Zustand übergeht. Beispielsweise kann in einer Prozessanlage, in welcher der erste SMFB einen Dampfkessel regelt, eine Ausgabe TRANS_OUT des ersten SMFB, die angibt, dass der erste SMFB von einem abgeschalteten Zustand in einen Anheizzustand übergeht, bewirken, dass ein zweiter SMFB, der ein Kesselentlüftungsgebläse regelt, während eines Zeitraums, bevor der erste SMFB den Kessel startet, von einem abgeschalteten Gebläsezustand in einen laufenden Gebläsezustand übergeht. Folglich wird ein eventueller Stau von explosiven Gasen im Innern des Kessels entlüftet, bevor der Kessel angeheizt wird.

[0054] **Fig.** 6 ist ein Beispiel der Matrix **300**, bei der Konfigurationsdaten in einigen Zellen eingetragen sind. Beispielsweise umfasst die Zelle **303A** Konfigurationsdaten, die den nächsten Zustand angeben, in den Zustandsmaschine übergehen soll, wenn sich die Zustandsmaschine in dem Zustand „AUSGELÖST" befindet, und wenn die Eingabe „ZURÜCKSETZEN ERLAUBNIS" bestätigt ist. Insbesondere geben die Konfigurationsdaten der Zelle **303A** an, dass die Zustandsmaschine in den Zustand „BEREIT ZUM ZURÜCKSETZEN" übergehen soll. Ferner umfasst die Zelle **302A** Konfigurationsdaten, die der „AKTION A" entsprechen, die der SMFB ausführen soll, wenn er in den Zustand „BEREIT ZUM ZURÜCKSETZEN" übergeht, und die Zelle **304A** umfasst Konfigurationsdaten, die der „AKTION B" entsprechen, die der SMFB ausführen soll, wenn er aus dem Zustand „AUSGELÖST" übergeht. Ähnlich umfasst die Zelle **303B** Konfigurationsdaten, die angeben, dass die Zustandsmaschine in den Zustand „WARTEN AUF START" übergehen soll (von „BEREIT ZUM ZURÜCKSETZEN"), wenn die Eingabe „RESET" bestätigt ist, die Zelle **302B** umfasst Konfigurationsdaten, die der „AKTION C" entsprechen, welche die Zustandsmaschine ausführen soll, wenn sie in den Zustand „WARTEN AUF START" übergeht, und die Zelle **304B** umfasst Konfigurationsdaten, die der „AKTION D" entsprechen, welche die Zustandsmaschine ausführen soll, wenn sie von dem Zustand „BEREIT ZUM ZURÜCKSETZEN" übergeht. Es versteht sich, dass AKTION A, AKTION B, AKTION C,

AKTION D usw. einer beliebigen Funktion oder Aktion entsprechen können, die von dem SMFB oder anderen Komponenten ausführbar ist, wie hier besprochen.

[0055] Zusätzlich kann eine bestimmte Zelle oder Gruppe von Zellen eine Übergang-in-Aktion ohne Übergang-aus-Aktion oder umgekehrt umfassen. Beispielsweise umfasst die Zelle **303C** Konfigurationsdaten, die angeben, dass die Zustandsmaschine in den Zustand „AUSGELÖST" (von „WARTEN AUF START") übergehen soll, wenn die Eingabe „AUSLÖSUNG ANGEFRAGT" bestätigt ist, und die Zelle **302C** umfasst Konfigurationsdaten, die der „AKTION E" entsprechen, welche die Zustandsmaschine ausführen soll, wenn sie in den Zustand „AUSGELÖST" übergeht. Die Zelle **303C** verfügt jedoch nicht über ein entsprechendes Konfigurationselement TRANS_OUT. Falls sich demnach die Zustandsmaschine in dem Zustand „WARTEN AUF START" befindet und die Eingabe „AUSLÖSUNG ANGEFRAGT" bestätigt ist, kann der SMFB die „AKTION E" gemäß dem Übergang von „WARTEN AUF START" auf „AUSGELÖST" ausführen. Ähnlich kann ein Eingabe/Zustands-Paar (wie etwa das in der Zelle **303D**) auch kein entsprechendes Konfigurationselement TRANS_IN oder TRANS_OUT aufweisen. Entsprechend kann die Zustandsmaschine von einem aktuellen Zustand in einen nächsten Zustand übergehen, ohne ein Konfigurationselement entweder TRANS_IN oder TRANS_OUT auszuführen.

[0056] Falls der Programmierer bei einigen Ausführungsformen keine Konfigurationsdaten in eine Zelle einträgt, kann man davon ausgehen, dass für diesen bestimmten Zustand und die Eingabe kein Zustandsübergang erfolgen soll. Beispielsweise umfassen die Zellen **302E**, **303E** und **304E** keine Konfigurationsdaten, die angeben, dass wenn sich die Zustandsmaschine in dem Zustand „AUSGELÖST" befindet, und wenn die Eingabe „WIEDERHERSTELLUNG STARTEN" bestätigt ist, die Zustandsmaschine in dem Zustand „AUSGELÖST" bleiben soll und keine Aktionen ausführen soll. Bei anderen Ausführungsformen kann der Programmierer Konfigurationsdaten eintragen, die angeben, dass die Zustandsmaschine für diese bestimmte Zustand/Eingabe-Kombination ihre Zustände nicht ändern soll.

[0057] Der Programmierer kann Konfigurationsdaten in die Matrix **300** unter Verwendung einer beliebigen von diversen Techniken eintragen, wozu Techniken gehören, die dem Fachmann wohlbekannt sind. Um beispielsweise Konfigurationsdaten in eine Zelle einzutragen, kann der Programmierer die Zelle unter Verwendung einer Maus, einer Steuerkugel, eines Berührungsbildschirms usw. auswählen. Dann könnte der Benutzer beispielsweise über eine Tastatur oder ein anderes Eingabegerät Konfigurationsdaten direkt in die Zelle eintragen. Alternativ könnte der Programmierer die Zelle auswählen und dann eine Auswahl „Bearbeiten", „Ändern" usw. aus einem Aktionsmenü treffen oder eine Schaltfläche „Bearbeiten", „Ändern" usw. auswählen. Dann kann die Benutzerschnittstelle dem Programmierer eine Liste von Zuständen über ein Aktionsmenü, ein Fenster, einen Bildschirm usw. vorlegen. Wahlweise kann die Zustandsliste den Zustand, dem die Zelle entspricht, oder eine Auswahl „KEIN ÜBERGANG" umfassen. Anschließend kann der Programmierer beispielsweise unter Verwendung einer Tastatur, einer Maus, einer Steuerkugel, eines Berührungsbildschirms usw. einen der Zustände auswählen. Falls der Programmierer den Zustand auswählt, welcher der Zelle oder der Auswahl „KEIN ÜBERGANG" entspricht, würden die Konfigurationsdaten angeben, dass für diese Kombination aus Zustand und Eingabe kein Übergang erfolgen soll.

[0058] Das Konfigurieren des SMFB unter Verwendung einer Benutzerschnittstelle, die eine Matrix umfasst, wie etwa die Matrix **300**, kann das Umsetzen einer Zustandsmaschine im Vergleich beispielsweise mit einem sequenziellen Funktionsdiagramm oder einer Programmiersprache, wie etwa C++, erleichtern. Beispielsweise würde das Umsetzen einer Zustandsmaschine unter Verwendung eines Programms in C++ wahrscheinlich zuerst das Erstellen eines Zustandsübergangsdiagramms mit verknüpften Übergangsaktionen und dann das Schreiben eines Programms, um das Diagramm umzusetzen, bedingen. Dann müsste das Programm getestet und auf Fehler untersucht werden, manchmal bevor es in einem Prozessregelungssystem umgesetzt wird. Mit einem SMFB, der unter Verwendung einer Matrix, wie etwa der Matrix **300**, konfiguriert ist, ist es jedoch nicht notwendig, ein Programm zu schreiben. Stattdessen würde das „Programmieren" einfach das Ausfüllen der Matrix mit den Zuständen und den damit verknüpften Übergangsaktionen bedingen. Weil zusätzlich kein Software-Code geschrieben werden muss, ist kein Debugging und Testen des Codes notwendig. Stattdessen kann das Testen einfach das Testen diverser Kombinationen von Zuständen, Übergang-in-Aktionen, Übergang-aus-Aktionen und Eingaben bedingen, um zu überprüfen, dass der SMFB in die richtigen nächsten Zustände übergeht und die gewünschten Übergangsaktionen ausführt. In manchen Fällen ist die Funktionsweise des SMFB einfach zu verstehen, indem man einfach die Matrix **300** kontrolliert. Somit könnte die Funktionsweise eines konfigurierten SMFB ohne Weiteres dokumentiert werden, indem beispielsweise eine Darstellung der Matrix gedruckt wird.

[0059] Ein SMFB, der gemäß einer Matrix, wie etwa der Matrix **300**, konfiguriert ist, kann beispielsweise in einem Sicherheitssystem oder einem Prozessregelungssystem verwendet werden. Als nur ein Beispiel kann ein SMFB, der gemäß einer Matrix, wie etwa der

Matrix **300**, konfiguriert ist, als Teil eines Sicherheitssystems verwendet werden, um einen Brenner in einer Prozessanlage zu verwalten. Beispielweise könnte der SMFB Zustände umfassen, wie etwa „ANHEIZEN", „GAS ABSCHALTEN" und „LÜFTEN". Wenn der Brenner hochgefahren wird, könnte der SMFB zuerst in den Zustand LÜFTEN gehen, um zu bewirken, dass eventuelles Gas in dem Brenner entlüftet wird. Dann könnte der SMFB in den Zustand ANHEIZEN gehen, um den Brenner anzuheizen. Auch falls die Flamme des Brenners erlischt, könnte der SMFB in den Zustand GAS ABSCHALTEN gehen, um das Gas zum Brenner abzuschalten. Dann könnte der SMFB in den Zustand LÜFTEN gehen. Zusätzlich kann der SMFB Übergang-in- und Übergang-aus-Aktionen umfassen, um das Sicherheitssystem zu ermöglichen. Beispielsweise kann der SMFB eine Übergang-in-Aktion umfassen, die mit dem Übergang in den Zustand LÜFTEN verknüpft ist, wodurch die Übergang-in-Aktion einen Bedienerindikator aktiviert, um einen Bediener darüber zu informieren, dass der Brenner entlüftet wird. Zusätzlich kann der SMFB eine Übergang-aus-Aktion umfassen, die mit dem Übergang aus dem Zustand LÜFTEN verknüpft ist, wodurch die Übergang-aus-Aktion den Bediener darüber informiert, dass der Brenner entsprechend entlüftet wird. Alternativ kann die Ausgabe TRANS_OUT oder TRANS_IN eines ersten SMFB (z. B. eines SMFB zur Kesselregelung) als eine Eingabe für einen anderen SMFB (z. B. einen SMFB zur Regelung einer Kesselbelüftungsanlage) oder ein anderes Prozessregelgerät bereitgestellt werden, um zu bewirken, dass der andere SMFB oder das andere Prozessregelgerät eine Funktion ausführt, während der erste SMFB einen Übergang aus dem aktuellen Zustand und/oder in einen nächsten Zustand vornimmt.

[0060] Ein SMFB, der gemäß einer Matrix, wie etwa der Matrix **300**, konfiguriert ist, kann von einem oder mehreren der Regler **12a**, **16a**, der E/A-Geräte **24**, der Logic Solver **50** und der Geräte **22**, **23**, **60**, **62** umgesetzt werden. Bei einigen Ausführungsformen kann der SMFB durch einen Prozessor, der gemäß einer Software konfiguriert ist, durch einen programmierbaren Logikbaustein, z. B. ein Gerät, das ein oder mehrere von einem Gate-Array, einer Standardzelle, einem anwenderprogrammierbaren Gate-Array (FPGA), einem PROM, einem EPROM, einem EEPROM, einer programmierbaren Array-Logik (PAL), eines programmierbaren logischen Arrays (PLA) usw. umfasst, umgesetzt werden.

[0061] Die Konfigurationsdaten, die mit einem SMFB verknüpft sind (beispielsweise Daten, die in eine Matrix, wie etwa die Matrix **300**, eingetragen werden, und wahlweise andere Konfigurationsdaten), können auf einem computerlesbaren Medium gespeichert werden, wie etwa auf einer Festplatte, einem RAM, einem ROM, einer CD-ROM, einem EPROM, einem EEPROM, einer DVD, einem FLASH-Speicher usw. und/oder einem Speicher, der mit einem Prozessor verknüpft ist.

[0062] **Fig.** 7 ist ein Ablaufschema eines beispielhaften Verfahrens **350** der Funktionsweise eines konfigurierten SMFB. Das Verfahren **350** kann regelmäßig und/oder beispielsweise als Reaktion auf ein Auslöseereignis umgesetzt werden. In einem Block **354** empfängt der SMFB seine Dateneingabe(n). Mit Bezug auf **Fig.** 4 empfängt der SMFB beispielsweise eine der Eingaben IN_D1 bis IN_D7. In einem Block **356** führt der SMFB eine geeignete Übergang-aus-Aktion basierend auf dem aktuellen Zustand und/oder der oder den Dateneingabe(n) aus. Insbesondere kann der SMFB die Übergang-aus-Aktion unter Verwendung der verknüpften Übergang-aus-Konfigurationsdaten ausführen, die in einer Konfigurationsdatenbank gespeichert sind. In einem Block **357** führt der SMFB eine geeignete Übergang-in-Aktion basierend auf dem nächsten Zustand aus (wie durch den aktuellen Zustand und die Dateneingabe(n) angegeben). Insbesondere kann der SMFB die Übergang-in-Aktion unter Verwendung der verknüpften Daten zum Konfigurieren eines Übergangs-in, die in einer Konfigurationsdatenbank gespeichert sind, ausführen.

[0063] In einem Block **358** ändert der SMFB gegebenenfalls einen Zustand seiner Zustandsmaschine basierend auf der oder den Dateneingabe(n), dem aktuellen Zustand des SMFB und den Konfigurationsdaten, die in einer Konfigurationsdatenbank gespeichert sind. Beispielsweise stellt der SMFB den aktuellen Zustand als den bestimmten nächsten Zustand ein. Die Daten der Konfigurationsdatenbank können Daten umfassen, die über eine Matrix, wie etwa die Matrix **300**, eingetragen werden. Der Zustand kann auch basierend auf anderen Faktoren geändert werden. Wie es beispielsweise nachstehend ausführlicher beschrieben wird, kann der SMFB konfiguriert sein, um eine oder mehrere der Dateneingaben zu ignorieren. Somit kann das Ändern des Zustands auch auf Konfigurationsdaten basieren, die angeben, welche der Dateneingabe(n) gegebenenfalls zu ignorieren sind. Als ein anderes Beispiel können zwei oder mehrere Dateneingaben angeben, dass eine Zustandsänderung von einem aktuellen Zustand in zwei oder mehrere nächste Zustände erfolgen soll. Somit kann der SMFB eine der Dateneingaben auswählen, um zu bestimmen, in welchen der möglichen nächsten Zustände der SMFB übergehen soll, basierend auf Prioritätsdaten, welche die Dateneingaben priorisieren. Als noch ein anderes Beispiel kann bzw. können die Dateneingabe(n) für den SMFB einen Status umfassen (z. B. einen Status RICHTIG oder einen Status FALSCH). Somit kann das Ändern des Zustands beispielsweise auch auf Konfigurationsdaten basieren, die angeben, wie eine Eingabe mit einem Status FALSCH zu handhaben ist.

[0064] Dann kann der SMFB in einem Block **366** seine Datenausgaben basierend auf dem aktuellen Zustand der Zustandsmaschine einstellen. Beispielsweise kann der SMFB die Ausgabe STATE (sowie eine geeignete Ausgabe OUT_Dx) auf den aktuellen Zustand der Zustandsmaschine einstellen. Ferner kann der SMFB die Ausgabe TRANS_OUT einstellen, um eine Übergang-aus-Aktion anzugeben, die auszuführen ist, wenn die Zustandsmaschine einen Übergang aus dem aktuellen Zustand vornimmt, und kann die Ausgabe TRANS_IN einstellen, um eine Übergang-in-Aktion anzugeben, die auszuführen ist, wenn die Zustandsmaschine in einen nächsten Zustand übergeht.

[0065] Noch einmal mit Bezug auf **Fig.** 4 kann der SMFB wahlweise eine Eingabe „ENABLE" umfassen. Falls bei einer Ausführungsform die Eingabe ENABLE nicht mehr bestätigt ist, kann der SMFB in einen gesperrten Zustand (z. B. den Zustand 0) gezwungen werden und muss in diesem Zustand bleiben, bis die Eingabe ENABLE bestätigt wird. Wenn die Eingabe ENABLE bestätigt wird, kann der SMFB in einen Anfangszustand (z. B. den Zustand 1) gezwungen werden, nach dem der SMFB gemäß den Konfigurationsdaten, die in eine Konfigurationsmatrix, wie etwa die Matrix **300** aus **Fig.** 5, eingetragen wurden, in andere Zustände übergehen kann.

[0066] Der SMFB kann zusätzlich eine Eingabe oder mehrere Eingaben umfassen, um die Zustandsmaschine in einen gewünschten Zustand zu zwingen. Beispielsweise umfasst der SMFB **200** eine Eingabe TRK_IN_D und eine Eingabe TRK_VAL. Wenn die Eingabe TRK_IN_D bestätigt wird, kann der SMFB in einen Zustand gezwungen werden, der durch die Eingabe TRK_VAL vorgegeben wird. Falls beispielweise die Eingabe TRK_VAL gleich „6" ist und die Eingabe TRK_IN_D bestätigt wird, kann der SMFB in den Zustand „6" gezwungen werden.

[0067] Wahlweise gibt es für das Konfigurieren des SMFB zusätzliche Möglichkeiten. Beispielsweise kann der SMFB eine Eingabe-(oder Übergangs-)Maske umfassen, die angibt, ob gegebenenfalls eine oder mehrere der Eingaben IN_D1, IN_D2 usw. zu ignorieren ist. Auch kann der SMFB konfiguriert sein, um auf Eingaben zu reagieren, die eine Vielzahl von Status aufweisen können. Beispielsweise kann eine oder können alle der Eingaben für den SMFB einen Status „richtig" oder einen Status „falsch" aufweisen, und der SMFB kann konfiguriert sein, um in Abhängigkeit von dem Status einer Eingabe unterschiedlich zu reagieren. Bei einem bestimmten Beispiel kann der SMFB konfiguriert sein, um eine Eingabe zu ignorieren, die „falsch" ist, die Eingabe zu verwenden, auch wenn sie „falsch" ist, oder den letzten „guten" Wert der Eingabe zu verwenden. Ferner kann der SMFB einen Parameter RESET umfassen, der, wenn er wahr ist, den SMFB in den Zustand „1" zwingt.

[0068] Die zuvor beschriebenen diversen Konfigurationsdaten und die Konfigurationsdaten des nächsten Zustands können auf dem gleichen computerlesbaren Medium oder auf verschiedenen computerlesbaren Medien gespeichert sein.

[0069] **Fig.** 8 ist ein Blockdiagramm eines Beispiels eines SMFB **400**. Der SMFB **400** umfasst eine Logik **404**, die einen nächsten Zustand bestimmt, der mindestens teilweise auf den Eingaben IN_D1, IN_D2 usw. und dem aktuellen Zustand des SMFB **400** basiert. Insbesondere greift die Logik **404** auf die Daten zur Konfiguration des nächsten Zustands zu, die in einer Datenbank zur Konfiguration des nächsten Zustands **406** gespeichert sind. Die Logik **404** bestimmt auch Funktionen von Übergang-in- und Übergang-aus-Aktionen, die von dem SMFB **400** auszuführen sind. Die Übergang-in-Aktion kann mindestens teilweise auf den Übergang-in-Konfigurationsdaten aus einer Datenbank zum Konfigurieren des Übergangs-aus **405** basieren. Die Übergang-aus-Aktion kann mindestens teilweise auf Daten zum Konfigurieren des Übergangs-aus aus einer Datenbank zum Konfigurieren des Übergangs-aus **407** basieren. Gemäß den Ausführungsformen, wie sie hier besprochen werden, können die Daten zum Konfigurieren des Übergangs-in Funktionen definieren, die der SMFB ausführt, bevor sich der SMFB in einem nächsten Zustand stabilisiert, und die Daten zum Konfigurieren des Übergangs-aus können Funktionen definieren, die der SMFB ausführt, bevor er aus einem aktuellen Zustand übergeht (oder ansonsten vor dem Übergang in einen nächsten Zustand). Die Datenbanken **405**, **406**, **407** können auf einem computerlesbaren Medium gespeichert sein, etwa wie es hier beschrieben wird. Die Daten zum Konfigurieren des nächsten Zustands, des Übergangs-in und des Übergangs-aus können Konfigurationsdaten umfassen, die in eine Matrix, wie etwa in die Matrix **300** aus **Fig.** 5, eingetragen werden. Es versteht sich, dass diverse Komponenten, Logikelemente oder Module die Daten zum Konfigurieren des Übergangs-in und/oder des Übergangs-aus ausführen können. Beispielsweise kann der SMFB **400** die Daten zum Konfigurieren des Übergangs-in und/oder des Übergangs-aus ausführen. Als weiteres Beispiel kann der SMFB **400** Befehle an eine getrennte Komponente oder ein getrenntes Modul senden, um die Daten zum Konfigurieren des Übergangs-in und/oder des Übergangs-aus auszuführen.

[0070] Gemäß einigen Ausführungsformen wird die Ausgabe der Logik **404** der Schaltlogik **408** bereitgestellt. Die Schaltlogik **408** wählt zwischen der Ausgabe der Logik **404** und der Eingabe TRK_VAL basierend auf der Eingabe TRK_IN_D aus. Falls beispielsweise die Eingabe TRK_IN_D bestätigt wird, kann die

Schaltlogik **408** die Eingabe TRK_VAL auswählen. Ansonsten kann die Schaltlogik **408** die Ausgabe der Logik **404** auswählen.

[0071] Die Ausgabe der Schaltlogik **408** wird der Schaltlogik **412** bereitgestellt, die basierend auf der Ausgabe der Freigabe- und Rücksetzlogik **416** zwischen der Ausgabe der Schaltlogik **408**, dem Wert 0 und dem Wert 1 auswählt. Die Ausgabe der Freigabe- und Rücksetzlogik **416** gibt an, ob der Zustand in einen gesperrten Zustand (Zustand 0) oder einen Anfangszustand (Zustand 1) zu zwingen ist. Die Freigabe- und Rücksetzlogik **416** generiert diese Ausgabe basierend auf der Eingabe ENABLE. Falls beispielsweise die Bestätigung der Eingabe ENABLE aufgehoben wird, kann die Ausgabe der Freigabe- und Rücksetzlogik **416** angeben, dass der Zustand auf 0 zu zwingen ist. Falls sich die Eingabe ENABLE von unbestätigt auf bestätigt ändert, kann die Ausgabe der Freigabe- und Rücksetzlogik **416** angeben, dass der Zustand auf 1 zu zwingen ist. Falls ENABLE bestätigt wird und zuvor bestätigt war, kann die Ausgabe der Freigabe- und Rücksetzlogik **416** angeben, dass der Zustand nicht auf 0 oder 1 gezwungen werden soll.

[0072] Die Ausgabe der Schaltlogik **412** ist der aktuelle Zustand des SMFB **400** und kann als Ausgabe des SMFB **400** bereitgestellt werden. Die Ausgabe der Schaltlogik **412** kann auch der Logik **420** bereitgestellt werden, die eine entsprechende Ausgabe OUT_D1, OUT_D2, TRANS_IN, TRANS_OUT usw. einstellt, die dem aktuellen Zustand des SMFB entspricht. Wie in **Fig.** 8 abgebildet, kann die Logik **420** optional auf Zustands-/Ausgabe-Konfigurationsdaten zugreifen, die in einer optionalen Ausgabekonfigurationsdatenbank **458** gespeichert sind. Die Datenbank **458** und die Datenbank **406** können auf demselben computerlesbaren Medium oder auf verschiedenen computerlesbaren Medien gespeichert sein. Die Ausgabekonfigurationsdaten können Konfigurationsdaten umfassen, die in eine Matrix, wie etwa die Matrix **700** aus **Fig.** 13, eingetragen sind, wie es hier besprochen wird.

[0073] Jeder der Blöcke **404**, **408**, **412**, **416** und **420** kann von einem oder mehreren von Hardware, Software und Firmware umgesetzt werden. Zusätzlich können einige der Blöcke kombiniert, umgestellt, geändert oder ausgelassen werden, und es können zusätzliche Blöcke hinzugefügt werden. Rein beispielhaft könnten die Blöcke **408** und **412** zu einem einzigen Block kombiniert werden.

[0074] **Fig.** 9 ist ein Ablaufschema eines Verfahrens **450** der Funktionsweise des beispielhaften SMFB **400**. Das Verfahren **450** aus **Fig.** 9 kann beispielsweise regelmäßig und/oder bei einem Auslöseereignis umgesetzt werden. In einem Block **454** werden die Dateneingaben des SMFB **400** verarbeitet. Beispielsweise kann bestimmt werden, ob einige der Dateneingaben IN_D1, IN_D2 usw. bestätigt wurden. Falls eine oder mehrere der Dateneingaben einen Status „FALSCH" aufweist, kann als anderes Beispiel bestimmt werden, wie eine oder mehrere „FALSCHE" Eingaben zu handhaben sind. In einem Block **458** wird die Eingabe ENABLE des SMFB **400** verarbeitet. Beispielsweise kann bestimmt werden, ob die Eingabe ENABLE bestätigt wird und/oder ob sie sich geändert hat, seit sie zuvor verarbeitet wurde.

[0075] In einem Block **459** führt der SMFB eine geeignete Übergang-aus-Aktion basierend auf dem aktuellen Zustand und/oder den Dateneingaben aus, indem er beispielsweise auf die verknüpften Daten zum Konfigurieren des Übergangs-aus zugreift, die in einer Konfigurationsdatenbank gespeichert sind. In einem Block **460** führt der SMFB eine geeignete Übergang-in-Aktion basierend auf dem nächsten Zustand und/oder den Dateneingaben aus, indem er beispielsweise auf die verknüpften Daten zum Konfigurieren des Übergangs-in zugreift, die in einer Konfigurationsdatenbank gespeichert sind. Beispielsweise kann der SMFB konfiguriert sein, um eine Einrichtung innerhalb einer Prozessanlage zu regeln. Wenn sich die Einrichtung in einem abgeschalteten Zustand (d. h. dem aktuellen Zustand) befindet, können die Daten zum Konfigurieren des nächsten Zustands ausgeführt werden, wenn eine Einrichtungsstart-Eingabe aktiviert wird und die Einrichtung in einen Betriebszustand (d. h. den nächsten Zustand) übergeht. Bevor er tatsächlich aus dem abgeschalteten Zustand übergeht, kann der SMFB die Daten zum Konfigurieren des Übergangs-aus ausführen, um einen Alarm zu aktivieren, der angibt, dass die Einrichtung im Begriff ist zu starten. Zusätzlich kann der SMFB, bevor er sich in dem Betriebszustand stabilisiert, die Daten zum Konfigurieren des Übergangs-in ausführen, um den Alarm auszuschalten.

[0076] Als weiteres Beispiel kann der SMFB konfiguriert sein, um mehrere Heizeinrichtungen in einer Lagerhalle zu regeln. Wenn sich die Prozessanlage in einem Heizzustand (d. h. dem aktuellen Zustand) befindet, kann eine Eingabe bestätigt werden, um einen Ventilator zu starten, was einem Übergang in einen Lüftungszustand entspricht. Während des Übergangs aus dem Heizzustand kann der SMFB automatisch einen Ofen oder eine andere Heizquelle sperren, um zu verhindern, dass die Heizquelle weiter Wärme generiert. Entsprechend wird der Ofen daran gehindert, nach dem Übergang in den Lüftungszustand zu funktionieren. Ferner kann der SMFB vor dem Übergang in den Lüftungszustand automatisch eine Reihe von Lüftungslöchern aktivieren, die mit dem Ventilator verknüpft sind, um zum Belüften der entsprechenden Maschine beizutragen.

[0077] In einem Block **462** kann ein Zustand des SMFB **400** bei Bedarf geändert werden. Zusätzlich

kann bzw. können eine oder mehrere Datenausgaben des SMFB **400** bei Bedarf geändert oder eingestellt werden. Beispielsweise kann bestimmt werden, dass eine Änderung der Dateneingaben angibt, dass der Zustand des SMFB **400** zu ändern ist. Falls sich zusätzlich der Zustand ändert, kann es sein, dass eine oder mehrere Datenausgaben, wie etwa die Ausgaben TRANS_IN und TRANS_OUT, des SMFB **400** zu ändern sind.

[0078] Mehrere beispielhafte Routinen, die man verwenden kann, um das Verfahren **450** mindestens teilweise umzusetzen, werden nun beschrieben. Beispielsweise ist **Fig.** 10 ein Ablaufschema einer beispielhaften Routine **500**, die verwendet werden kann, um die Dateneingaben IN_D1, IN_D2 usw. in den SMFB zu verarbeiten. In einem Block **504** wird eine Variable z auf eins gesetzt. In einem Block **508** wird bestimmt, ob der Status der Dateneingabe IN_Dz „FALSCH" ist. Falls der Status nicht falsch ist, dann wird das Bit Nummer z einer Variablen TRANSITIONS auf den Wert der Dateneingabe IN_Dz gesetzt. Falls der Status falsch ist, kann dann bestimmt werden, wie die Dateneingabe zu handhaben ist. Bei einem Beispiel kann der SMFB die Eingaben „FALSCH" auf drei Arten handhaben: die Eingabe FALSCH kann trotzdem verwendet werden (IMMER_VERWENDEN), sie kann ignoriert werden (IGNORIEREN_WENN_FALSCH), oder der letzte Eingabewert „RICHTIG" kann verwendet werden (LETZTEN_RICHTIGEN_VERWENDEN). Somit kann in einem Block **516** bestimmt werden, ob der SMFB die letzte Dateneingabe „RICHTIG" verwenden soll. Falls der SMFB den letzten Wert „RICHTIG" verwenden soll, dann kann man den Block **512** überspringen. Ansonsten kann dann in Block **520** bestimmt werden, ob der SMFB den Eingabewert FALSCH ignorieren soll. Falls der SMFB den Wert FALSCH nicht ignorieren soll, kann die Routine dann mit Block **512** fortfahren. Falls der SMFB den Wert FALSCH ignorieren soll, kann die Routine dann mit einem Block **524** fortfahren. In Block **524** wird das Bit Nummer „z" der Variablen TRANSITIONS auf 0 gesetzt.

[0079] In einem Block **528** wird die Variable z inkrementiert, und in einem Block **532** kann bestimmt werden, ob die Variable z größer als die Anzahl der Dateneingaben in den SMFB ist. Falls z nicht größer als die Anzahl der Dateneingaben in den SMFB ist, kann die Routine zu Block **508** zurückkehren, um die nächste Dateneingabe zu verarbeiten. Ansonsten kann die Routine enden.

[0080] Fig. 11 ist ein Ablaufschema einer beispielhaften Routine **545**, die man verwenden kann, um die Eingabe ENABLE für den SMFB zu verarbeiten. In einem Block **550** kann bestimmt werden, ob ein Wert einer Variable LASTENABLE der gleiche Wert ist wie die Eingabe ENABLE. Die Variable LASTENABLE gibt im Allgemeinen den Wert von ENABLE zu einem vorhergehenden Zeitpunkt an (beispielsweise den Wert der Variablen ENABLE während des vorhergehenden Ablaufs der Routine **545**). Falls die Werte von LASTENABLE und ENABLE gleich sind, kann die Routine **545** enden. Ansonsten kann die Routine mit einem Block **554** fortfahren, in dem bestimmt werden kann, ob die Eingabe ENABLE bestätigt ist. Falls die Eingabe ENABLE bestätigt ist, kann in einem Block **558** eine Variable RESET auf WAHR gesetzt werden.

[0081] Falls in dem Block **554** bestimmt wird, dass die Eingabe ENABLE nicht bestätigt ist, dann wird in Block **562** die Bestätigung der Ausgabe OUT_D1, OUT_D2 usw., die dem aktuellen Wert einer Variablen STATE entspricht, aufgehoben. In einem Block **566** wird dann die Variable STATE auf 0 gesetzt. Nach den Blöcken **558** und **566** kann die Routine mit einem Block **570** fortfahren, in dem die Variable LASTENABLE auf den Wert der Eingabe ENABLE gesetzt wird. Nach dem Block **570** kann die Routine enden.

[0082] Fig. 12 ist ein Ablaufschema einer beispielhaften Routine **600**, die man verwenden kann, um einen nächsten Zustand des SMFB zu bestimmen und um bei Bedarf eine geeignete Ausgabe OUT_D1, OUT_D2, TRANS_IN, TRANS_OUT usw. einzustellen. In einem Block **604** kann bestimmt werden, ob die Eingabe ENABLE bestätigt ist. Falls nicht, kann die Routine enden. Falls die Eingabe ENABLE bestätigt ist, kann die Routine mit einem Block **608** fortfahren, in dem eine Variable NEWSTATE auf 0 gesetzt wird. Dann kann in einem Block **612** bestimmt werden, ob die Eingabe TRK_IN_D bestätigt ist. Wenn sie bestätigt ist, kann die Routine mit einem Block **616** fortfahren, in dem die Variable NEWSTATE auf den Wert der Eingabe TRK_VAL gesetzt wird.

[0083] Falls in Block **612** bestimmt wird, dass die Eingabe TRK_IN_D nicht bestätigt ist, kann die Routine mit einem Block **620** fortfahren. In Block **620** kann bestimmt werden, ob die Variable RESET WAHR ist. Wenn ja, kann die Routine mit einem Block **624** fortfahren, in dem die Variable NEWSTATE auf 1 gesetzt werden kann. In einem Block **626** kann dann die Variable RESET auf FALSCH gesetzt werden.

[0084] Falls in dem Block **620** bestimmt wird, dass die Variable RESET nicht WAHR ist, dann kann die Routine mit einem Block **632** fortfahren. In dem Block **632** kann eine Variable TEMP dadurch bestimmt werden, dass Bit für Bit eine Variable TRANSITION_MASK, die Variable TRANSITIONS und ein Element eines Arrays STATECHANGEMASK, auf das die Variable STATE weist, mit „AND" addiert werden. Die Variable TRANSITION_MASK kann eine konfigurierbare Variable sein, die man verwenden kann, um zu verhindern, dass gewisse Eingaben IN_Dx das Vorkommen einer Zustandsänderung verursachen. Falls

beispielsweise ein Programmierer verhindern möchte, dass die Eingabe IN_D3 verursacht, dass sich der Zustand der Zustandsmaschine ändert, könnte der Programmierer das dritte Bit der Variablen TRANSITION_MASK auf 0 setzen. Wenn der Programmierer zulassen möchte, dass die Eingabe D3 bewirkt, dass sich der Zustand der Zustandsmaschine ändert, könnte der Programmierer das dritte Bit der Variablen TRANSITION_MASK auf 1 setzen.

[0085] Jedes Element des Arrays STATECHANGEMASK kann eine Variable sein, die für einen entsprechenden der Zustände angibt, welche Eingaben IN_D1, IN_D2 usw. eine Zustandsänderung bewirken. Insbesondere kann jedes Element des Arrays einem der Zustände der Zustandsmaschine entsprechen. Beispielsweise kann STATECHANGEMASK[1] dem Zustand 1 entsprechen, STATECHANGEMASK [2] kann dem Zustand 2 entsprechen, usw. Zusätzlich kann jedes Bit von jedem Element einer der Eingaben IN_D1, IN_D2 usw. entsprechen. Beispielsweise kann das Bit 1 IN_D1 entsprechen, das Bit 2 kann IN_D2 entsprechen usw. Beispielsweise mit Bezug auf **Fig.** 6 hätte das Array STATECHANGEMASK für die Matrix **300** 6 Elemente und das Element STATECHANGEMASK[3] wäre 0×44.

[0086] Nach dem Block **628** kann die Routine mit einem Block **632** fortfahren, in dem bestimmt werden kann, ob die Variable TEMP gleich 0 ist. Wenn sie nicht gleich 0 ist, kann die Routine mit einem Block **636** fortfahren, in dem eine Variable z auf die Zahl des ersten Bits (d. h. ausgehend von dem niedrigstwertigen Bit) in der Variablen TEMP gesetzt werden kann, das nicht gleich null ist. Dies stellt nämlich die Prioritäten der Eingaben basierend auf ihrer Reihenfolge derart ein, dass IN_D1 die höchste Priorität ist, IN_D2 die nächsthöhere Priorität ist, IN_D3 die nächsthöhere Priorität ist, usw. Bei anderen Ausführungsformen könnten andere Priorisierungsmethoden verwendet werden. Beispielsweise könnte man es einem Programmierer erlauben, den Eingaben Prioritäten zuzuweisen, oder es könnte eine andere Prioritätsreihenfolge verwendet werden (z. B. ist IN_D1 die niedrigste Priorität, IN_D2 ist die nächstniedrigere Priorität usw.). Die Prioritäten könnten für den SMFB insgesamt oder für jeden Zustand eingestellt werden. Dann kann in einem Block **640** die Variable NEWSTATE auf den Wert der Zustandsübergangsmatrix in der Reihe z und der Spalte ZUSTAND gesetzt werden.

[0087] Nach den Blöcken **616**, **626** und **640** kann die Routine mit einem Block **644** fortfahren. Falls in dem Block **632** bestimmt wird, dass die Variable TEMP gleich 0 ist, kann die Routine auch mit dem Block **644** fortfahren. In dem Block **644** kann bestimmt werden, ob die Variable NEWSTATE gleich 0 ist. Wenn sie gleich 0 ist, kann die Routine enden. Wenn sie nicht gleich 0 ist, kann die Routine mit einem Block **645** fortfahren, in dem die Übergang-aus-Aktion ausgeführt werden kann. Dann kann in einem Block **646** die Übergang-in-Aktion ausgeführt werden. In einem Block **648** wird die Bestätigung der Ausgaben OUT_D1, OUT_D2, TRANS_IN, TRANS_OUT usw., die der Variablen STATE entsprechen, aufgehoben. Es versteht sich, dass die Ausgaben, deren Bestätigung aufzuheben ist, nicht unbedingt der Variablen STATE entsprechen müssen. In einem Block **652** wird die Variable STATE auf den Wert der Variable NEWSTATE gesetzt. In einem Block **656** werden die Ausgaben OUT_D1, OUT_D2, TRANS_IN, TRANS_OUT usw., die der Variablen STATE entsprechen, bestätigt und die Routine kann enden. Es versteht sich, dass die zu bestätigenden Ausgaben nicht unbedingt der Variablen STATE entsprechen müssen.

[0088] Es versteht sich, dass das Verfahren **450** aus **Fig.** 9 und die Routinen aus **Fig.** 10 bis **Fig.** 12 rein beispielhaft sind und dass die Blöcke bei anderen Beispielen geändert werden können, neue Blöcke hinzugefügt werden können, die Blöcke umgestellt werden können, Blöcke ausgelassen werden können und/oder die Blöcke kombiniert werden können. Mit Bezug auf **Fig.** 10 können rein beispielhaft die Blöcke **508**, **516**, **520** und **524** ausgelassen werden, wenn keine spezielle Handhabung der Eingaben mit einem Status „FALSCH” benötigt oder erwünscht ist.

[0089] Als anderes Beispiel könnte der Block **636** derart geändert werden, dass die Variable z auf die Zahl des letzten Bits in TEMP, das nicht gleich 0 ist, gesetzt wird. Als noch ein anderes Beispiel könnte der Block **636** geändert werden, um z auf die Zahl zu setzen, die einem der Bits in TEMP entspricht, das nicht gleich 0 ist, basierend auf gewissen Prioritätsdaten.

[0090] Noch einmal mit Bezug auf **Fig.** 4 sind nicht alle Datenausgaben unbedingt Indikatoren des Zustands der Zustandsmaschine, die dem SMFB **200** entspricht. Beispielsweise können bei einer Ausführungsform die Werte für die Ausgaben OUT_D1, OUT_D2 usw., die diversen Zuständen der Zustandsmaschine entsprechen, konfigurierbar sein. Somit kann beispielsweise für bestimmte Zustände eine Vielzahl der Ausgaben OUT_D1, OUT_D2 usw. bestätigt werden. Damit ein Programmierer den SMFB konfigurieren kann, kann eine Konfigurationsanwendung auf dem Anzeigegerät **120** einen Benutzerschnittstellen-Mechanismus anzeigen, wie etwa ein Konfigurationsfenster, einen Bildschirm usw., der mit dem Funktionsblock verknüpft ist.

[0091] **Fig.** 13 ist ein Beispiel eines Benutzerschnittstellen-Mechanismus, der verwendet werden kann, um mindestens teilweise einen SMFB zu konfigurieren, wie etwa den SMFB **200** aus **Fig.** 4. Der Benutzerschnittstellen-Mechanismus umfasst eine Tabelle oder Matrix **700** (nachstehend als „Matrix **700**” be-

zeichnet), die als Teil eines Konfigurationsfensters, eines Bildschirms usw., das bzw. der mit dem SMFB verknüpft ist, angezeigt werden kann. Die Matrix **700** umfasst eine Vielzahl von Zellen **704**, die in Reihen und Spalten angeordnet sind. Jede Spalte entspricht einer Ausgabe von einer Vielzahl von Ausgaben OUT_D1, OUT_D2 usw., TRANS_IN und TRANS_OUT des Zustandsmaschinen-Funktionsblocks, und jede Reihe entspricht einem der möglichen Zustände der Zustandsmaschine. Somit entspricht jede Zelle **704** einem Zustand und einer Ausgabe. Bei anderen Beispielen kann jede Spalte einer der Vielzahl von Ausgaben entsprechen, und jede Spalte kann einem der möglichen Zustände der Zustandsmaschine entsprechen.

[0092] Die Ausgaben „1" bis „4" der Matrix **700** können jeweils den Ausgaben OUT_D1 bis OUT_D4 des SMFB entsprechen, und die Ausgaben „5" und „6" können jeweils den Ausgaben TRANS_IN und TRANS_OUT entsprechen. Ähnlich können die Zustände „1" bis „6" der Matrix **700** den möglichen Zuständen der Zustandsmaschine entsprechen. Zusätzlich kann ein Benutzer bei diesem Beispiel in der Lage sein, jede der Ausgaben zu bezeichnen. Beispielsweise ist in **Fig.** 13 „Ausgabe 1" mit „VENTIL VLV-101 ÖFFNEN" bezeichnet. Das Bezeichnen der Ausgaben kann dazu beitragen, das Verständnis der Funktionsweise der Zustandsmaschine und/oder das Erstellen einer Schnittstelle zwischen der Zustandsmaschine und der Prozessanlage zu erleichtern.

[0093] Ein Programmierer kann den SMFB konfigurieren, indem er Konfigurationsinformationen in die Zellen **704** einträgt. Insbesondere kann der Programmierer für eine bestimmte Zelle **704**, die einem der Zustände und einer der Ausgaben entspricht, Konfigurationsdaten in die Zelle eintragen, die angeben, dass wenn sich die Zustandsmaschine in diesem Zustand befindet, die Ausgabe zu bestätigen ist. Bei der beispielhaften Matrix **700** wurden Konfigurationsdaten in einige der Zellen **704** eingetragen. Beispielsweise umfasst die Zelle **704A** Konfigurationsdaten, die angeben, dass wenn sich die Zustandsmaschine in dem Zustand „AUSGELÖST" befindet, die Ausgabe OUT_D3 zu bestätigen ist, die Zelle **704B** umfasst Konfigurationsdaten, die angeben, dass wenn sich die Zustandsmaschine ist dem Zustand „AUSGELÖST" befindet, die Ausgabe TRANS_IN nicht zu bestätigen ist, und die Zelle **704C** umfasst Konfigurationsdaten, die angeben, dass wenn sich die Zustandsmaschine in dem Zustand „WIEDERHERGESTELLT" befindet, die Ausgabe TRANS_OUT zu bestätigen ist.

[0094] Wenn der Programmierer bei diesem bestimmten Beispiel keine Konfigurationsdaten in eine Zelle **704** einträgt, kann man davon ausgehen, dass für diesen bestimmten Zustand die entsprechende Ausgabe nicht zu bestätigen ist. Beispielsweise umfassen die Zellen **704D** und **704E** kein X, was angibt, dass wenn sich die Zustandsmaschine in dem Zustand „AUSGELÖST" befindet, die Ausgaben OUT_D1 und OUT_D2 nicht zu bestätigen sind. Bei anderen Ausführungsformen kann der Programmierer Konfigurationsdaten eintragen, die angeben, dass die Zustandsmaschine bestimmte Ausgaben nicht bestätigen soll, falls sie sich in einem bestimmten Zustand befindet. Ähnlich kann es möglich sein anzugeben, dass es für einen bestimmten Zustand und eine bestimmte Ausgabe unwichtig ist, ob die Ausgabe bestätigt ist oder nicht.

[0095] Der Programmierer kann unter Verwendung einer von diversen Techniken, einschließlich der Techniken, die dem Fachmann wohlbekannt sind, Konfigurationsdaten in die Matrix **700** eintragen. Um beispielsweise Konfigurationsdaten in eine Zelle **704** einzutragen, kann der Programmierer die Zelle **704** unter Verwendung einer Maus, einer Steuerkugel, eines Berührungsbildschirms usw. auswählen. Dann könnte der Benutzer, beispielsweise über eine Tastatur, Konfigurationsdaten direkt in die Zelle **704** eintragen. Alternativ könnte der Programmierer die Zelle **704** auswählen und dann eine Auswahl „Bearbeiten", „Ändern" usw. aus einem Aktionsmenü treffen oder eine Schaltfläche „Bearbeiten", eine Schaltfläche „Ändern" usw. auswählen. Dann kann die Benutzerschnittstelle für den Programmierer eine Auswahlliste über ein Aktionsmenü, ein Fenster, einen Bildschirm usw. anzeigen. Beispielsweise kann die Auswahlliste eine Auswahl „Ausgabe bestätigen", eine Auswahl „Bestätigung der Ausgabe aufheben" und wahlweise eine Auswahl „unwichtig" umfassen. Dann kann der Programmierer, beispielsweise unter Verwendung einer Tastatur, einer Maus, einer Steuerkugel, eines Berührungsbildschirms usw., eine der Auswahlmöglichkeiten auswählen. Falls der Programmierer die Auswahl „Ausgabe bestätigen" trifft, können die Konfigurationsdaten angeben, dass für den entsprechenden Zustand die entsprechende Ausgabe zu bestätigen ist. Beispielsweise kann ein „X" in der Zelle angezeigt werden, eine „1" kann in der Zelle angezeigt werden, das Wort „WAHR" kann in der Zelle angezeigt werden, das Wort „BESTÄTIGEN" kann in der Zelle angezeigt werden, usw. Falls der Programmierer die Auswahl „Bestätigung der Ausgabe aufheben" trifft, können die Konfigurationsdaten angeben, dass für den entsprechenden Zustand die entsprechende Ausgabe nicht zu bestätigen ist. Beispielsweise kann die Zelle leer bleiben, eine „0" kann in der Zelle angezeigt werden, das Wort „FALSCH" kann in der Zelle angezeigt werden, die Wörter „BESTÄTIGUNG AUFHEBEN" können in der Zelle angezeigt werden usw.

[0096] Obwohl die beispielhafte Matrix **700** Reihen für sechs Zustände und sechs Ausgaben umfasst, können ähnliche Matrizen mit anderen Anzahlen von Zuständen und Ausgaben für SMFB verwendet wer-

den, die andere Anzahlen von Zuständen und Ausgaben aufweisen. Die Anzahl von Zuständen und Ausgaben kann konfigurierbar sein.

[0097] Noch einmal mit Bezug auf **Fig.** 7, und wie zuvor beschrieben, können die Datenausgaben des SMFB, nachdem der aktuelle Zustand bestimmt wurde, basierend auf dem aktuellen Zustand eingestellt werden (Block **366**). Beispielsweise können die Datenausgaben gemäß den Konfigurationsdaten eingestellt werden, die in eine Matrix, wie etwa in die Matrix **700** aus **Fig.** 13, eingetragen wurden.

[0098] **Fig.** 14 ist ein Ablaufschema einer beispielhaften Routine **850**, die verwendet werden kann, um die geeigneten Ausgaben OUTD1_D1, OUT_D2 usw. zu bestätigen. In einem Block **854** wird eine Variable z auf eins gesetzt. In einem Block **858** wird die Ausgabe OUT_Dz auf den Wert der Bitanzahl z eines Elements einer Array-Variablen OUTPUT eingestellt, auf welche die Variable STATE zeigt. Jedes Element des Arrays OUTPUT kann eine Variable sein, die für einen entsprechenden der Zustände die Werte der Ausgaben OUT_D1, OUT_D2 usw. angibt. Beispielsweise kann OUTPUT[1] einem Zustand 1 entsprechen, OUTPUT[2] kann einem Zustand 2 entsprechen, usw. Zusätzlich kann jedes Bit von jedem Element einer der Ausgaben OUT_D1, OUT_D2 usw. entsprechen. Beispielsweise kann das Bit 1 OUT_D1 entsprechen, das Bit 2 kann OUT_D2 entsprechen, usw. Mit Bezug auf **Fig.** 13 würde das Array OUTPUT beispielsweise für die Matrix **700** 6 Elemente aufweisen, und das Element OUTPUT[1] kann 0 × 06 sein.

[0099] In einem Block **862** wird die Variable z inkrementiert, und in einem Block **866** kann bestimmt werden, ob der Wert von z größer als die Anzahl der Ausgaben OUT_D1, OUT_D2 usw. ist. Falls z nicht größer als die Anzahl der Ausgaben OUT_D1, OUT_D2 usw. ist, kann die Routine zu Block **858** zurückkehren. Andernfalls kann die Routine enden.

[0100] Die Konfigurationsdaten für den SMFB können über andere Arten von grafischen Benutzerschnittstellen zusätzlich zu den zuvor beschriebenen eingetragen werden. Beispielsweise können Konfigurationsdaten über eine grafische Benutzerschnittstelle, die ähnlich wie ein Zustandsübergangsdiagramm ist, eingetragen werden. **Fig.** 15 ist ein beispielhaftes Zustandsübergangsdiagramm **900**, das man verwenden könnte, um einen SMFB, wie etwa den SMFB **200**, zu konfigurieren, wie mit Bezug auf **Fig.** 4 beschrieben. Das Diagramm **900** umfasst eine Vielzahl von grafischen Elementen **904**, **908**, **912**, **916**, **920**, **924**, **928** und **932**. Die Elemente **904**, **908** und **912** stellen jeweils die Zustände 1, 2, und 3 einer Zustandsmaschine dar. Das Element **916** gibt an, dass wenn sich die Zustandsmaschine in Zustand 1 befindet, sie in Zustand 2 übergehen soll, falls INPUT 2 bestätigt ist. Das Element **920** gibt an, dass wenn sich die Zustandsmaschine in Zustand 1 befindet, sie in Zustand 3 übergehen soll, falls INPUT 3 bestätigt ist. Das Element **924** gibt an, dass wenn sich die Zustandsmaschine in Zustand 2 befindet, sie in Zustand 1 übergehen soll, falls INPUT 1 bestätigt ist, und das Element **928** gibt an, dass wenn sich die Zustandsmaschine in Zustand 3 befindet, sie in Zustand 1 übergehen soll, falls INPUT 1 bestätigt ist. Ähnlich gibt das Element **932** an, dass wenn sich die Zustandsmaschine in Zustand 3 befindet, sie in Zustand 2 übergehen soll, falls INPUT 4 bestätigt wird.

[0101] Das Zustandsübergangsdiagramm **900** umfasst ferner die Übergang-in-Elemente **902**, **903** und ein Übergang-aus-Element **905**, die mit Zustand 1 verknüpft sind; die Übergang-in-Elemente **906**, **907** und ein Übergang-aus-Element **909**, die mit Zustand 2 verknüpft sind; und ein Übergang-in-Element **911** und die Übergang-aus-Elemente **913**, **914**, die mit Zustand 3 verknüpft sind. Falls sich insbesondere die Zustandsmaschine in Zustand 1 befindet und INPUT 3 bestätigt ist, soll die Zustandsmaschine das Übergang-aus-Element **905** ausführen, in den Zustand 3 übergehen und das Übergang-in-Element **911** von Zustand 3 ausführen. Falls sich ferner die Zustandsmaschine in Zustand 2 befindet und INPUT 1 bestätigt ist, soll die Zustandsmaschine das Übergang-aus-Element **909** ausführen, in den Zustand 1 übergehen und das Übergang-in-Element **902** ausführen. Es versteht sich, dass die Übergang-in-Elemente und/oder die Übergang-aus-Elemente für beliebige Zustände gemeinsam sein können. Beispielsweise können die Übergang-in-Elemente **902**, **903** gleich sein, wodurch die Zustandsmaschine die verknüpfte Übergang-in-Aktion als Reaktion darauf ausführt, dass die Zustandsmaschine entweder aus Zustand 2 oder Zustand 3 in den Zustand 1 übergeht. Als weiteres Beispiel können die Übergang-aus-Elements **913**, **914** gleich sein, wodurch die Zustandsmaschine die verknüpfte Übergang-aus-Aktion als Reaktion darauf ausführt, dass die Zustandsmaschine entweder in den Zustand 1 (als Reaktion darauf, dass INPUT 1 bestätigt ist) oder den Zustand 2 (als Reaktion darauf, dass INPUT 4 bestätigt ist) übergeht.

[0102] Im Allgemeinen kann ein SMFB durch Software, Firmware oder Hardware oder eine Kombination aus Software, Firmware und/oder Hardware umgesetzt werden kann. Beispielsweise kann ein SMFB durch einen oder mehrere der Regler **12a**, **16a**, der E/A-Geräte **24**, der Logic Solver **50** und der Geräte **22**, **23**, **60**, **62** umgesetzt werden. Als anderes Beispiel kann ein SMFB durch eine oder mehrere der Arbeitsstationen **18a** und **20a** umgesetzt werden. Beispielsweise kann der SMFB durch die Arbeitsstation **18a** und/oder die Arbeitsstation **20a** als Teil einer Simulation umgesetzt werden, um die Funktionsweise der Prozessanlage zu testen oder ein Bedienertraining bereitzustellen. Bei einigen Ausführungsformen kann der SMFB durch einen Prozessor, der ge-

mäß einer Software konfiguriert ist, durch einen programmierbaren Logikbaustein, z. B. ein Gerät, das ein oder mehrere von einem Gate-Array, einer Standardzelle, einem anwenderprogrammierbaren Gate-Array (FPGA), einem PROM, einem EPROM, einem EEPROM, einer programmierbaren Array-Logik (PAL), eines programmierbaren logischen Arrays (PLA) usw. umfasst, umgesetzt werden.

[0103] Jeder der Blöcke **404**, **408**, **412**, **416**, **420** und **458** aus **Fig.** 8 kann durch Software, Firmware oder Hardware oder eine Kombination von Software, Firmware und/oder Hardware umgesetzt werden. Obwohl zusätzlich die Ablaufschemata aus **Fig.** 10 bis **Fig.** 12 und **Fig.** 14 als Routinen beschrieben wurden, könnten diese Ablaufschemata durch Software, Hardware, Firmware oder eine Kombination aus Software, Firmware und/oder Hardware umgesetzt werden.

[0104] Ausführungsformen einer Benutzerschnittstelle, wie etwa der zuvor beschriebenen Benutzerschnittstellen, können insgesamt oder teilweise durch einen Prozessor umgesetzt werden, der beispielsweise gemäß einem Software-Programm konfiguriert ist. Beispielsweise kann die Arbeitsstation **18a** oder **20a** oder ein anderer Computer die zuvor beschriebene Benutzerschnittstelle vollständig oder teilweise umsetzen. Ein Software-Programm zum Umsetzen von Ausführungsformen einer Benutzerschnittstelle kann als Software auf einem materiellen Medium, wie etwa auf einer Festplatte, einem RAM, einem batteriegestützten RAM, einem ROM, einer CD-ROM, einem PROM, einem EPROM, einem EEPROM, einer DVD, einem Flash-Speicher usw. oder einem Speicher, wie etwa einem RAM, der mit dem Prozessor verknüpft ist, umgesetzt werden, doch der Fachmann wird ohne Weiteres verstehen, dass das gesamte Programm oder Teile desselben alternativ von einer anderen Vorrichtung als einem Prozessor ausgeführt werden könnte und/oder als Firmware und/oder dedizierte Hardware auf wohlbekannte Art und Weise ausgebildet sein könnte.

[0105] Ebenfalls hierin offenbart sind die folgenden Gegenstände:

1) Materielles Medium, das maschinenlesbare Anweisungen speichert, umfassend:
– ersten Code, um eine grafische Benutzerschnittstelle über ein Anzeigegerät zum Konfigurieren von Übergängen zwischen den Zustandsmaschinenzuständen einer Zustandsmaschine bereitzustellen, wobei die grafische Benutzerschnittstelle eine erste Vielzahl von Zellen angibt, die in einer Matrix angeordnet ist, die eine erste Dimension und eine zweite Dimension aufweist, wobei die Positionen entlang der ersten Dimension die Zustandsmaschinenzustände und die damit verknüpften Übergangsaktions-Kennungen angeben, und die Positionen entlang der zweiten Dimension Eingaben der Zustandsmaschine entsprechen, so dass die erste Vielzahl von Zellen Eingabe/Zustands-Paare und ihre Übergangsaktionen basierend auf den Positionen der ersten Vielzahl von Zellen mit Bezug auf die ersten und zweiten Dimensionen definiert;
– zweiten Code, um Zustandsübergangsdaten, die mit einer Zelle der ersten Vielzahl von Zellen verknüpft sind, über die grafische Benutzerschnittstelle zu empfangen, wobei die Zustandsübergangsdaten einen nächsten Zustand identifizieren, in den die Zustandsmaschine gemäß dem Eingabe/Zustands-Paar, das von der Zelle definiert wird, übergeht;
– dritten Code, um Übergangsaktionsdaten über die grafische Benutzerschnittstelle zu empfangen, wobei die Übergangsaktionsdaten mindestens eine Übergangsaktion identifizieren, die gemäß den Zustandsübergangsdaten auszuführen ist; und
– vierten Code, um auf einem computerlesbaren Medium die Zustandsübergangsdaten und die Übergangsaktionsdaten zu speichern, die mit einem Funktionsblock verknüpft sind, der die Zustandsmaschine in einer Prozessanlage umsetzt, so dass die Zustandsmaschine in den nächsten Zustand übergeht, wenn eine Bedingung in der Prozessanlage dem Eingabe/Zustands-Paar entspricht, das mit der Zelle verknüpft ist.

2) Materielles Medium nach Gegenstand 1) Anspruch 14, wobei die Übergangsaktionsdaten 1) eine Übergang-aus-Aktion, die beim Übergang aus dem Zustand der Zustandsmaschine, welcher der Zelle entspricht, auszuführen ist, und/oder 2) eine Übergang-in-Aktion, die beim Übergang in den nächsten Zustand auszuführen ist, identifiziert.

3) Materielles Medium nach Gegenstand 2), ferner umfassend:
– fünften Code, um die erste Vielzahl von Zellen auf dem Anzeigegerät anzuzeigen; und
– sechsten Code, um in der Zelle eine Angabe des nächsten Zustands, eine Angabe der Übergang-aus-Aktion und eine Angabe der Übergang-in-Aktion anzuzeigen.

4) Materielles Medium nach Gegenstand 3), wobei der sechste Code in der Zelle die Angabe des nächsten Zustands, die Angabe der Übergang-aus-Aktion und die Angabe der Übergang-in-Aktion anzeigt durch:
– Anzeigen der Angabe des nächsten Zustands in einer ersten Teilzelle der Zelle;
– Anzeigen der Angabe der Übergang-aus-Aktion in einer zweiten Teilzelle der Zelle; und
– Anzeigen der Angabe der Übergang-in-Aktion in einer dritten Teilzelle der Zelle.

5) Materielles Medium nach Gegenstand 3), wobei der fünfte Code die erste Vielzahl von Zellen auf dem Anzeigegerät durch Anzeigen der Matrix angibt, wobei die erste Dimension mindestens eine Zellenreihe aufführt und die zweite Dimen-

sion eine Vielzahl von Zellenspalten aufführt, so dass jede der mindestens einen Zellenreihe mit einer der Eingaben der Zustandsmaschine verknüpft ist, und jede der Vielzahl von Zellenspalten mit einem der Zustandsmaschinenzustände und mit einer der Übergangsaktions-Kennungen verknüpft ist.

6) Materielles Medium nach Gegenstand 3), wobei der fünfte Code die erste Vielzahl von Zellen auf dem Anzeigegerät durch Anzeigen der Matrix anzeigt, wobei die erste Dimension mindestens eine Zellenspalte anzeigt und die zweite Dimension eine Vielzahl von Zellenreihen anzeigt, so dass jede der Vielzahl von Zellenreihen mit einem der Zustandsmaschinenzustände und mit einer der Übergangsaktions-Kennungen verknüpft ist, und jede der mindestens einen Zellenspalte mit einer der Eingaben der Zustandsmaschine verknüpft ist.

7) Materielles Medium nach einem der Gegenstände 1) bis 6), wobei ein bestimmter Wert jeder der Eingaben der Zustandsmaschine eines von einer logischen Eins, einer logischen Null, einem logischen WAHR oder einem logischen FALSCH ist.

8) Materielles Medium nach einem der Gegenstände 1) bis 7), wobei die grafische Benutzerschnittstelle ferner eine zweite Vielzahl von Zellen angibt, die mit jedem Funktionsblock verknüpft ist, wobei jede der zweiten Vielzahl von Zellen einer jeweiligen Ausgabe von einer Vielzahl von Ausgaben des Funktionsblocks und einem jeweiligen der Zustandsmaschinenzustände entspricht, und wobei das materielle Medium ferner Folgendes umfasst:

– fünften Code, um Ausgabe-Konfigurationsdaten, die mit einer zusätzlichen Zelle der zweiten Vielzahl von Zellen verknüpft ist, über das Eingabegerät zu empfangen, wobei die Ausgabe-Konfigurationsdaten einen Ausgabewert, welcher der zusätzlichen Zelle entspricht, wenn sich die Zustandsmaschine in dem Zustandsmaschinen-Zustand befindet, welcher der zusätzlichen Zelle entspricht, und mindestens eine Ausgabeübergangsaktion, die mit der zusätzlichen Zelle verknüpft ist, angeben.

9) Materielles Medium nach einem der Gegenstände 1) bis 8), wobei die Eingaben der Zustandsmaschine mit mindestens einem von einem Prozessregelungssystem, einer Simulation eines Prozessregelungssystems, einem Sicherheitssystem und einer Simulation eines Sicherheitssystems verknüpft sind.

10) Materielles Medium nach einem der Gegenstände 1), 2), 7) und 9), ferner umfassend fünften Code zum Empfangen der Eingaben der Zustandsmaschine entweder von einem zusätzlichen Funktionsblock, der mit der Prozessanlage verknüpft ist, oder von einer Bedienerschnittstelle.

[0106] Obwohl die Erfindung zu diversen Änderungen und alternativen Konstruktionen fähig ist, wurden gewisse erläuternde Ausführungsformen derselben in den Zeichnungen gezeigt und werden hier ausführlich beschrieben. Es versteht sich jedoch, dass es keineswegs beabsichtigt ist, die Offenbarung auf die spezifischen offenbarten Formen einzuschränken, sondern dass es vielmehr beabsichtigt ist, dass die Erfindung alle Änderungen, alternativen Konstruktionen und Äquivalente abdeckt, die in Geist und Umfange der Offenbarung fallen, wie sie durch die beiliegenden Ansprüche definiert wird.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

Diese Liste der vom Anmelder aufgeführten Dokumente wurde automatisiert erzeugt und ist ausschließlich zur besseren Information des Lesers aufgenommen. Die Liste ist nicht Bestandteil der deutschen Patent- bzw. Gebrauchsmusteranmeldung. Das DPMA übernimmt keinerlei Haftung für etwaige Fehler oder Auslassungen.

Zitierte Patentliteratur

- US 7730415 [0004]

Patentansprüche

1. Verfahren zum Konfigurieren über ein Computergerät, das ein Anzeigegerät und ein Eingabegerät aufweist, eines Funktionsblocks, der mit einer Prozessanlage verknüpft ist, wobei der Funktionsblock dazu vorgesehen ist, eine Zustandsmaschine umzusetzen, wobei das Verfahren folgende Schritte umfasst:
– Bereitstellen einer grafischen Benutzerschnittstelle, die von dem Anzeigegerät angezeigt wird, wobei die grafische Benutzerschnittstelle eine erste Vielzahl von Zellen angibt, die mit dem Funktionsblock verknüpft ist und in einer Matrix angeordnet ist, die eine erste Dimension und eine zweite Dimension aufweist, wobei die Positionen entlang der ersten Dimension Zustände einer Zustandsmaschine und damit verknüpfte Kennungen von Übergangsaktionen angeben, und Positionen entlang der zweiten Dimension den Eingaben der Zustandsmaschine entsprechen, so dass die erste Vielzahl von Zellen Eingabe/Zustands-Paare und ihre Übergangsaktionen basierend auf den Positionen der ersten Vielzahl von Zellen mit Bezug auf die ersten und zweiten Dimensionen definieren;
– Empfangen von Zustandsübergangsdaten, die mit einer Zelle der ersten Vielzahl von Zellen über das Eingabegerät verknüpft sind, wobei die Zustandsübergangsdaten einen nächsten Zustand identifizieren, in den die Zustandsmaschine nach einer Bedingung in der Prozessanlage, die dem Eingabe/Zustands-Paar entspricht, das durch die Zelle definiert wird, übergeht;
– Empfangen von Übergangsaktionsdaten über das Eingabegerät, wobei die Übergangsaktionsdaten mindestens eine Übergangsaktion identifizieren, die gemäß den Zustandsübergangsdaten auszuführen ist; und
– Speichern der Zustandsübergangsdaten und der Übergangsaktionsdaten auf einem computerlesbaren Medium, das mit dem Funktionsblock verknüpft ist.

2. Verfahren nach Anspruch 1, wobei die Übergangsaktionsdaten mindestens eine identifizieren von 1) einer Übergang-aus-Aktion, die beim Übergang aus dem Zustand der Zustandsmaschine, welcher der Zelle entspricht, auszuführen ist, und 2) einer Übergang-in-Aktion, die beim Übergang in den nächsten Zustand auszuführen ist.

3. Verfahren nach Anspruch 1 oder 2, ferner umfassend folgende Schritte:
– Anzeigen der ersten Vielzahl von Zellen an dem Anzeigegerät; und
– Anzeigen einer Angabe des nächsten Zustands, einer Angabe der Übergang-aus-Aktion und einer Angabe der Übergang-in-Aktion in der Zelle.

4. Verfahren nach Anspruch 3, wobei das Anzeigen der Angabe des nächsten Zustands, der Angabe der Übergang-aus-Aktion und der Angabe des Übergang-in-Aktion in der Zelle folgende Schritte umfasst:
– Anzeigen der Angabe des nächsten Zustands in einer ersten Teilzelle der Zelle;
– Anzeigen der Angabe der Übergang-aus-Aktion in einer zweiten Teilzelle der Zelle; und
– Anzeigen der Angabe der Übergang-in-Aktion in einer dritten Teilzelle der Zelle.

5. Verfahren nach Anspruch 3, wobei das Anzeigen der ersten Vielzahl von Zellen auf dem Anzeigegerät das Anzeigen der Matrix umfasst, wobei die erste Dimension mindestens eine Reihe von Zellen aufführt und die zweite Dimension eine Vielzahl von Spalten von Zellen aufführt, so dass jede der mindestens einen Zellenreihe mit einer der Eingaben der Zustandsmaschinen verknüpft ist, und jede der Vielzahl von Zellenspalten mit einem der Zustände der Zustandsmaschine und mit einer der Kennungen der Übergangsaktion verknüpft ist.

6. Verfahren nach Anspruch 3, wobei das Anzeigen der ersten Vielzahl von Zellen auf dem Anzeigegerät das Anzeigen der Matrix umfasst, wobei die erste Dimension mindestens eine Zellenspalte aufführt und die zweite Dimension eine Vielzahl von Zellenreihen aufführt, so dass jede der Vielzahl von Zellenreihen mit einem der Zustände der Zustandsmaschine und mit einer der Kennungen der Übergangsaktion verknüpft ist und jede der mindestens einen Zellenspalte mit einer der Eingaben der Zustandsmaschine verknüpft ist.

7. Verfahren nach einem der vorhergehenden Ansprüche, wobei ein bestimmter Wert jeder der Eingaben der Zustandsmaschine eine logische Eins, eine logische Null, ein logisches WAHR oder ein logisches FALSCH ist.

8. Verfahren nach einem der vorhergehenden Ansprüche, wobei die grafische Benutzerschnittstelle ferner eine zweite Vielzahl von Zellen angibt, die mit dem Funktionsblock verknüpft ist, wobei jede der zweiten Vielzahl von Zellen einer jeweiligen Ausgabe von einer Vielzahl von Ausgaben des Funktionsblocks und einem jeweiligen Zustand von den Zuständen der Zustandsmaschine entspricht, und wobei das Verfahren ferner folgenden Schritt umfasst:
– Empfangen von Ausgabe-Konfigurationsdaten, die mit einer zusätzlichen Zelle der zweiten Vielzahl von Zellen über das Eingabegerät verknüpft sind, wobei die Ausgabe-Konfigurationsdaten einen Ausgabewert, welcher der zusätzlichen Zelle entspricht, wenn sich die Zustandsmaschine in dem Zustand der Zustandsmaschine befindet, welcher der zusätzlichen Zelle entspricht, und mindestens eine Ausgabe-Übergangsaktion, die mit der zusätzlichen Zelle verknüpft ist, angeben.

9. Verfahren nach einem der vorhergehenden Ansprüche, wobei die mindestens eine Übergangsaktion von dem Funktionsblock auszuführen ist.

10. Verfahren nach einem der vorhergehenden Ansprüche, ferner umfassend folgende Schritte:
– Empfangen von Prioritätsdaten, die mit Eingaben der Zustandsmaschine verknüpft sind; und
– Speichern der Prioritätsdaten, die mit den Eingaben der Zustandsmaschine verknüpft sind.

11. Verfahren nach einem der vorhergehenden Ansprüche, ferner umfassend folgende Schritte:
– Empfangen von Daten, die angeben, ob gegebenenfalls eine oder mehrere der Eingaben der Zustandsmaschine von der Zustandsmaschine ignoriert werden soll(en); und
– Speichern der Daten, die angeben, ob gegebenenfalls eine oder mehrere der Eingaben der Zustandsmaschine von der Zustandsmaschine ignoriert werden soll(en).

12. Verfahren nach einem der vorhergehenden Ansprüche, wobei die Eingaben der Zustandsmaschine mit mindestens einem von einem Prozessregelungssystem, einer Simulation eines Prozessregelungssystems, einem Sicherheitssystem und einer Simulation eines Sicherheitssystems verknüpft sind.

13. Verfahren nach einem der vorhergehenden Ansprüche, ferner umfassend das Empfangen der Eingaben der Zustandsmaschine von einem zusätzlichen Funktionsblock, der mit der Prozessanlage verknüpft ist oder von einer Bedienerschnittstelle.

14. Materieller Träger, der maschinenlesbare Anweisungen gespeichert hat, die, wenn sie auf einem Prozessor ausgeführt werden, ein Verfahren nach einem der Ansprüche 1–13 durchführen.

15. Verfahren zum Betreiben eines ersten Funktionsblocks, der mit einer Prozessanlage verknüpft ist, wobei der erste Funktionsblock eine Zustandsmaschine umsetzt, die eine Übergangstabelle und einen aktuellen Zustand aufweist, wobei das Verfahren folgende Schritte umfasst:
– Empfangen einer Eingabe für die Zustandsmaschine, wobei die Eingabe eine Bedingung innerhalb der Prozessanlage angibt;
– Kontrollieren der Übergangstabelle, um basierend auf dem aktuellen Zustand und der Eingabe einen nächsten Zustand und mindestens eine Übergangsaktion zu identifizieren;
– Einleiten durch den ersten Funktionsblock der mindestens einen Übergangsaktion;
– Einstellen des aktuellen Zustands der Zustandsmaschine auf den nächsten Zustand; und
– Bereitstellen einer Funktionsblockausgabe für einen zweiten Funktionsblock zur Verwendung beim Regeln eines Feldgeräts, wobei die Funktionsblockausgabe auf dem aktuellen Zustand der Zustandsmaschine basiert.

16. Verfahren nach Anspruch 15, wobei die mindestens eine Übergangsaktion eine Übergang-aus-Aktion und eine Übergang-in-Aktion vorgibt, und wobei das Einleiten der mindestens einen Übergangsaktion folgende Schritte umfasst:
– Einleiten der Übergang-aus-Aktion durch den ersten Funktionsblock; und
– Einleiten der Übergang-in-Aktion durch den ersten Funktionsblock.

17. Verfahren nach Anspruch 16, wobei der erste Funktionsblock die Übergang-aus-Aktion einleitet, bevor er die Übergang-in-Aktion einleitet.

18. Verfahren nach Anspruch 16, wobei der erste Funktionsblock die Übergang-in-Aktion einleitet, bevor er den aktuellen Zustand der Zustandsmaschine auf den nächsten Zustand einstellt.

19. Verfahren nach Anspruch 16, wobei der erste Funktionsblock die Übergang-aus-Aktion einleitet, bevor er den aktuellen Zustand der Zustandsmaschine auf den nächsten Zustand setzt.

20. Verfahren nach Anspruch 15–19, wobei das Einleiten der mindestens einen Übergangsaktion das Bereitstellen einer zusätzlichen Funktionsblockausgabe für einen dritten Funktionsblock umfasst, wobei die zusätzliche Funktionsblockausgabe die mindestens eine Übergangsaktion angibt.

21. Verfahren nach Anspruch 15–20, wobei das Einleiten der mindestens einen Übergangsaktion folgende Schritte umfasst:
– Abrufen aus einer Datenbank mindestens eines Übergangs-Konfigurationselements, das mit der mindestens einen Übergangsaktion verknüpft ist; und
– Ausführen des mindestens einen Übergangs-Konfigurationselements.

22. Verfahren nach Anspruch 15–20, ferner umfassend:
– als Reaktion darauf, dass der aktuelle Zustand auf den nächsten Zustand eingestellt wird, Einstellen mindestens einer Ausgabe, die mit dem aktuellen Zustand verknüpft ist.

23. Verfahren nach Anspruch 22, wobei das Einstellen der mindestens einen Ausgabe das Einstellen einer Zustandsangabe-Ausgabe und/oder einer Übergangsausgabe umfasst.

24. Zustandsmaschinen-Reglereinheit zur Verwendung in einer Prozessanlage, wobei die Zustandsmaschinen-Reglereinheit kommunikativ mit einem Feldgerät gekoppelt ist und eine Zustandsmaschine umsetzt, die eine Übergangstabelle und einen

aktuellen Zustand aufweist, und wobei die Zustandsmaschinen-Reglereinheit Folgendes umfasst:
– ein Eingabemodul, um eine Eingabe zu empfangen, die eine Bedingung innerhalb der Prozessanlage angibt; und
– ein Ausführungsmodul, das konfiguriert ist zum:
– Untersuchen der Übergangstabelle, um basierend auf dem aktuellen Zustand und der Eingabe einen nächsten Zustand und mindestens eine Übergangsaktion zu identifizieren,
– Einleiten der mindestens einen Übergangsaktion,
– Einstellen des aktuellen Zustands der Zustandsmaschine auf den nächsten Zustand, und
– Bereitstellen einer Ausgabe zur Verwendung beim Steuern des Feldgeräts, wobei die Ausgabe auf dem aktuellen Zustand der Zustandsmaschine basiert.

25. Zustandsmaschinen-Reglereinheit nach Anspruch 24, wobei die mindestens eine Übergangsaktion eine Übergang-aus-Aktion und eine Übergang-in-Aktion vorgibt, und wobei das Ausführungsmodul die mindestens eine Übergangsaktion dadurch einleitet, dass es die Übergang-aus-Aktion und die Übergang-in-Aktion einleitet.

26. Zustandsmaschinen-Reglereinheit nach Anspruch 25, wobei das Ausführungsmodul die Übergang-aus-Aktion einleitet, bevor es die Übergang-in-Aktion einleitet.

27. Zustandsmaschinen-Reglereinheit nach Anspruch 25, wobei das Ausführungsmodul die Übergang-in-Aktion einleitet, bevor es den aktuellen Zustand der Zustandsmaschine auf den nächsten Zustand einstellt.

28. Zustandsmaschinen-Reglereinheit nach Anspruch 25, wobei das Ausführungsmodul die Übergang-aus-Aktion einleitet, bevor es den aktuellen Zustand der Zustandsmaschine auf den nächsten Zustand einstellt.

29. Zustandsmaschinen-Reglereinheit nach einem der Ansprüche 24–28, wobei das Ausführungsmodul die mindestens eine Übergangsaktion durch Bereitstellen einer zusätzlichen Ausgabe für ein zusätzliches Modul einleitet, wobei die zusätzliche Ausgabe die mindestens eine Übergangsaktion angibt.

30. Zustandsmaschinen-Reglereinheit nach einem der Ansprüche 24–29, wobei das Ausführungsmodul die mindestens eine Übergangsaktion einleitet durch:
– Abrufen aus einer Datenbank mindestens eines Übergangs-Konfigurationselements, das mit der mindestens einen Übergangsaktion verknüpft ist, und
– Ausführen des mindestens einen Übergangs-Konfigurationselements.

31. Zustandsmaschinen-Reglereinheit nach einem der Ansprüche 24–30, wobei das Ausführungsmodul ferner konfiguriert ist, um mindestens eine Ausgabe einzustellen, die mit dem aktuellen Zustand verknüpft ist, als Reaktion auf das Einstellen des aktuellen Zustands auf den nächsten Zustand.

32. Zustandsmaschinen-Reglereinheit nach Anspruch 31, wobei das Ausführungsmodul die mindestens eine Ausgabe dadurch einstellt, dass es mindestens entweder eine Zustandsangabe-Ausgabe und/oder eine Übergangsausgabe einstellt.

Es folgen 14 Seiten Zeichnungen

Anhängende Zeichnungen

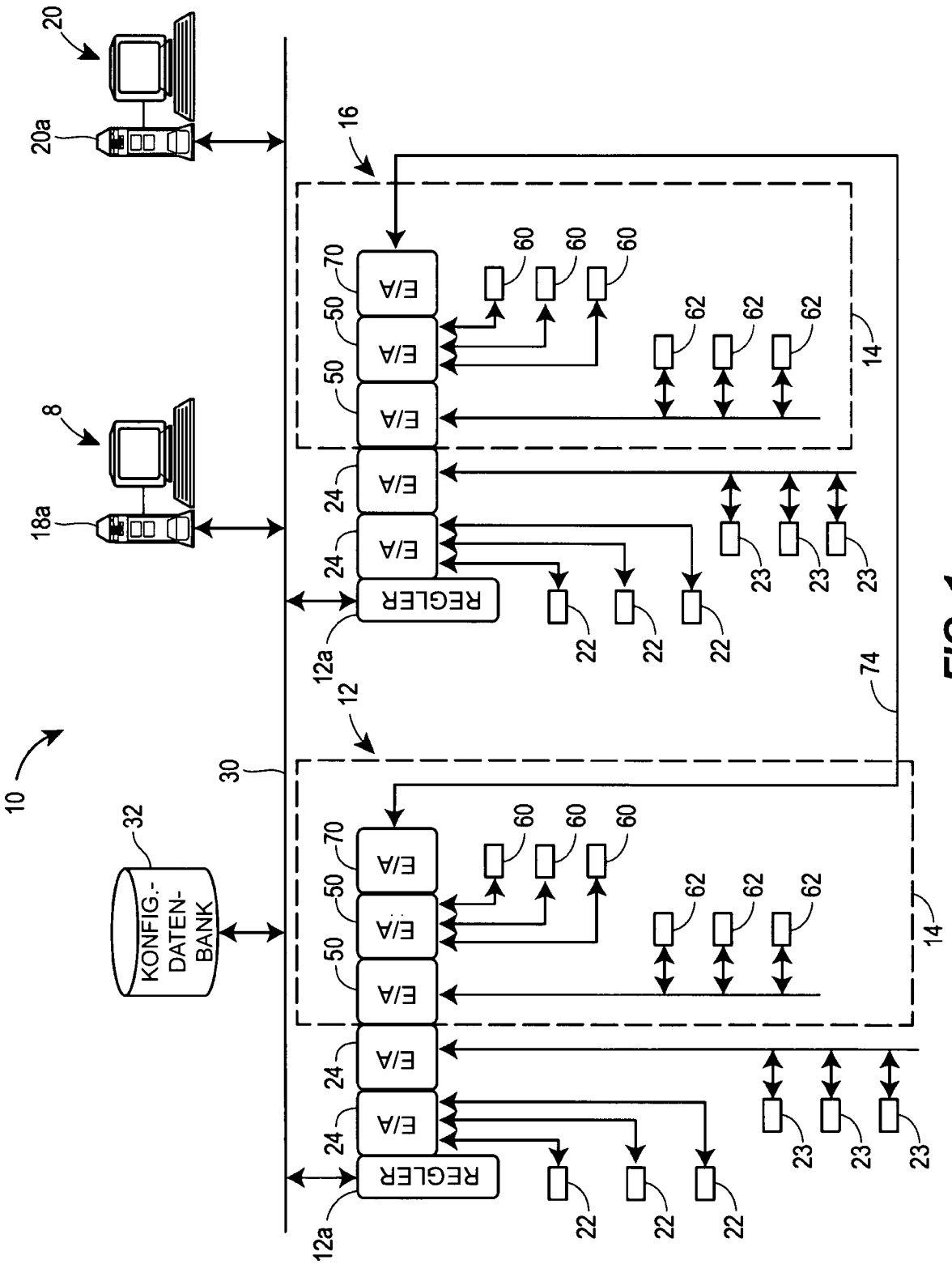


FIG. 1

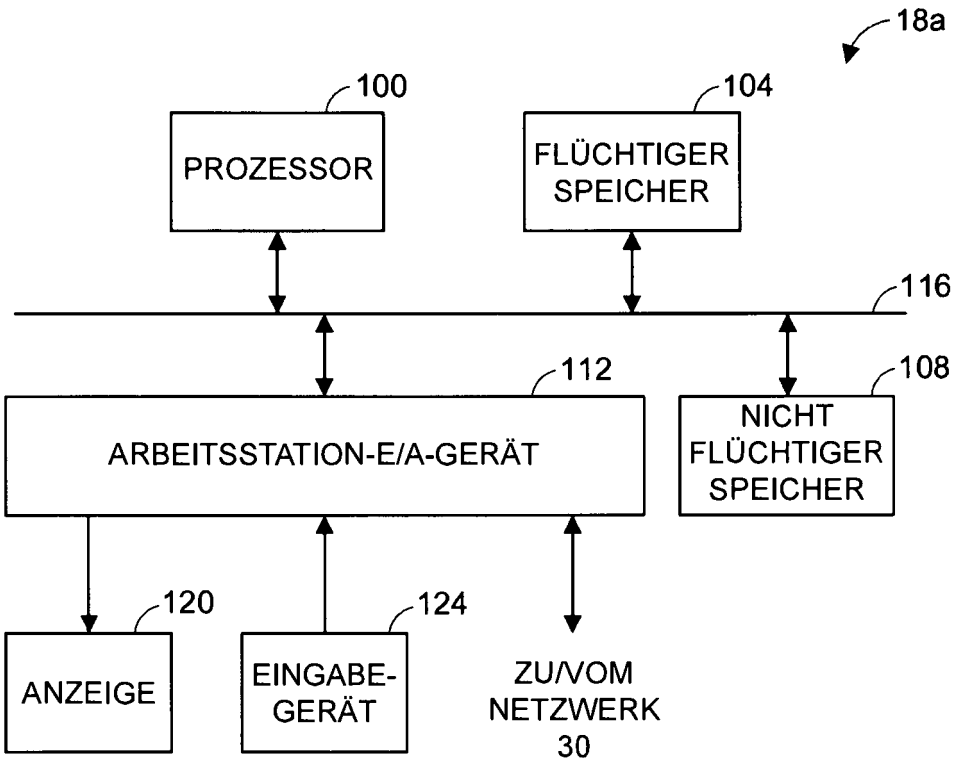

18a
100
PROZESSOR
104
FLÜCHTIGER SPEICHER
116
112
ARBEITSSTATION-E/A-GERÄT
108
NICHT FLÜCHTIGER SPEICHER
120
ANZEIGE
124
EINGABE-GERÄT
ZU/VOM NETZWERK 30

FIG. 2

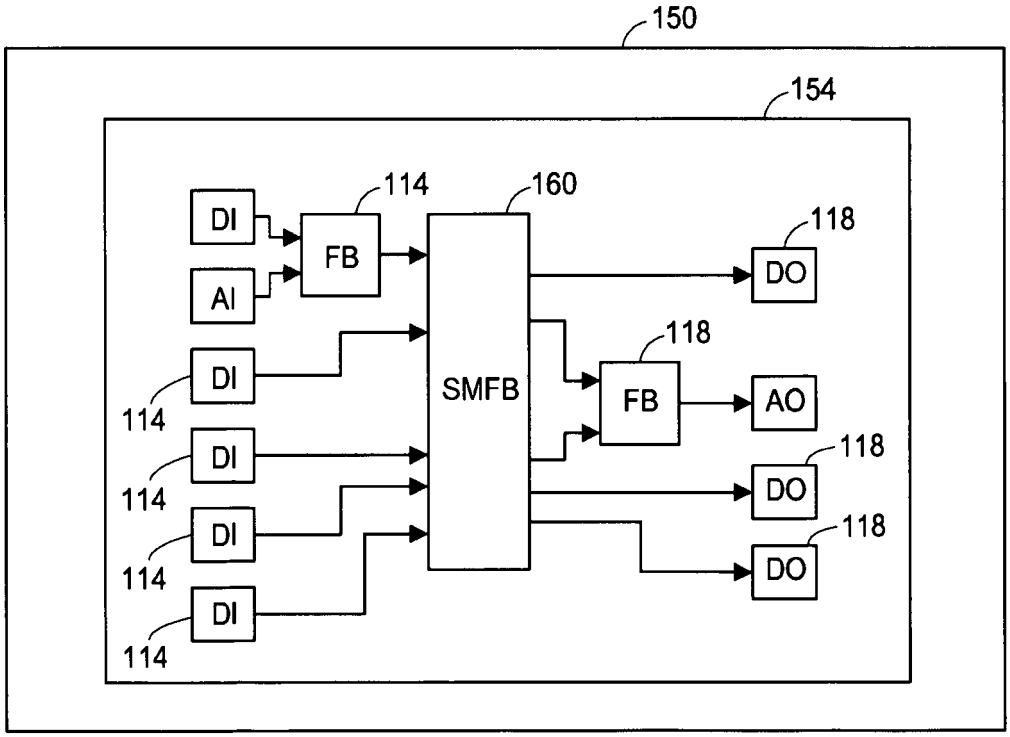

150
154
114
160
DI
AI
FB
SMFB
118
DO
118
FB
AO
118
DO
118
DO
114
DI
114
DI
114
DI
114
DI

FIG. 3

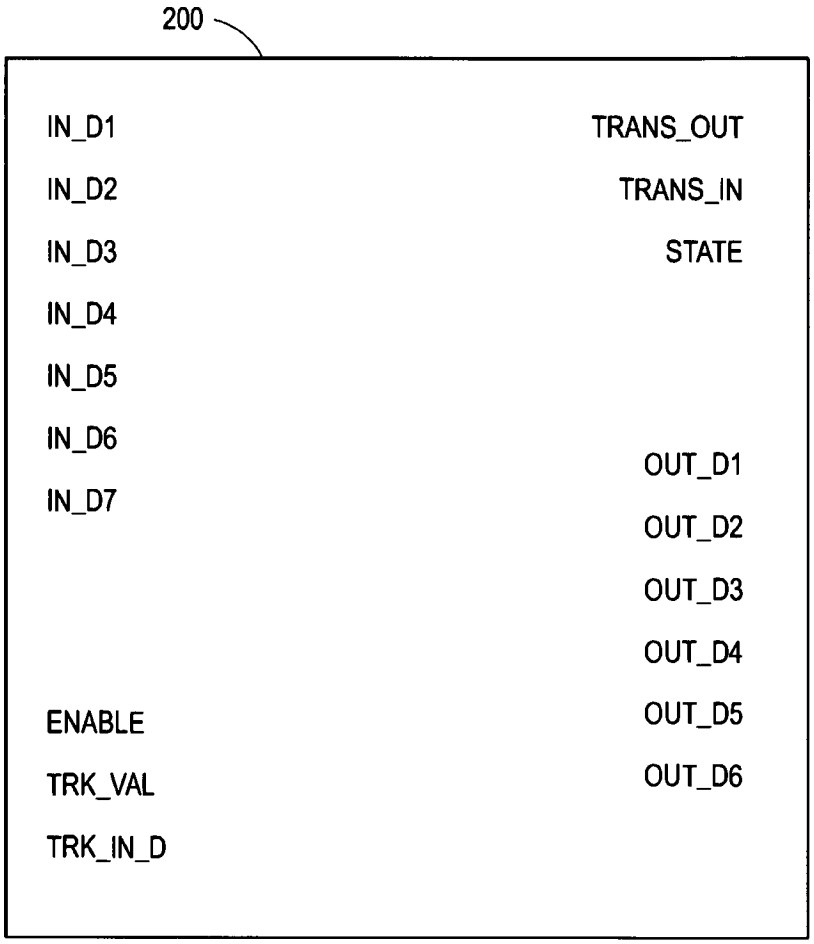

200
IN_D1
IN_D2
IN_D3
IN_D4
IN_D5
IN_D6
IN_D7
ENABLE
TRK_VAL
TRK_IN_D
TRANS_OUT
TRANS_IN
STATE
OUT_D1
OUT_D2
OUT_D3
OUT_D4
OUT_D5
OUT_D6

FIG. 4

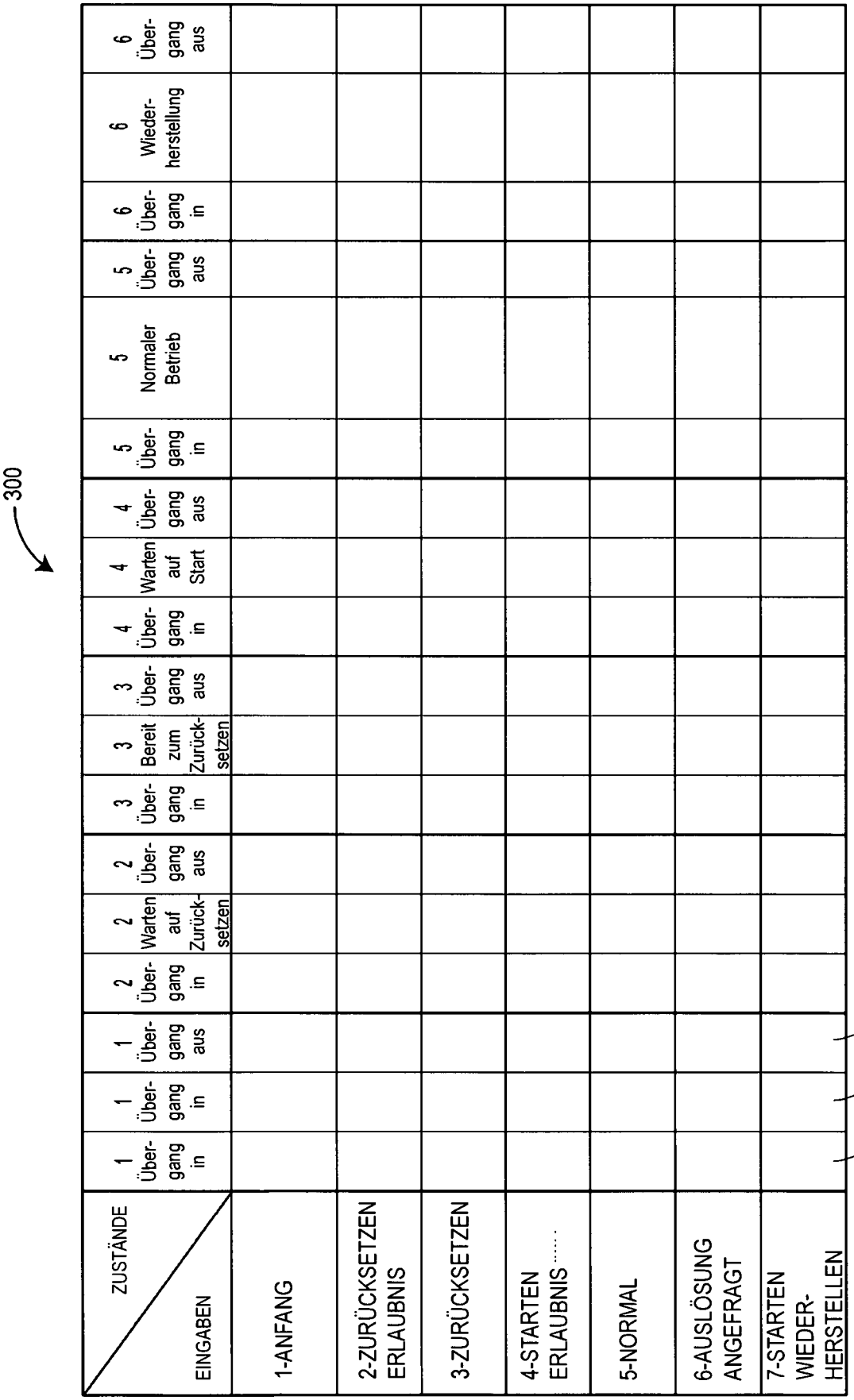

300
ZUSTÄNDE
EINGABEN
1-ANFANG
2-ZURÜCKSETZEN ERLAUBNIS
3-ZURÜCKSETZEN
4-STARTEN ERLAUBNIS
5-NORMAL
6-AUSLÖSUNG ANGEFRAGT
7-STARTEN WIEDER-HERSTELLEN
1 Über-gang in
1 Über-gang in
1 Über-gang aus
2 Über-gang in
2 Warten auf Zurück-setzen
2 Über-gang aus
3 Über-gang in
3 Bereit zum Zurück-setzen
3 Über-gang aus
4 Über-gang in
4 Warten auf Start
4 Über-gang aus
5 Über-gang in
5 Normaler Betrieb
5 Über-gang aus
6 Über-gang in
6 Wieder-herstellung
6 Über-gang aus
302
303
304 ...

FIG. 5

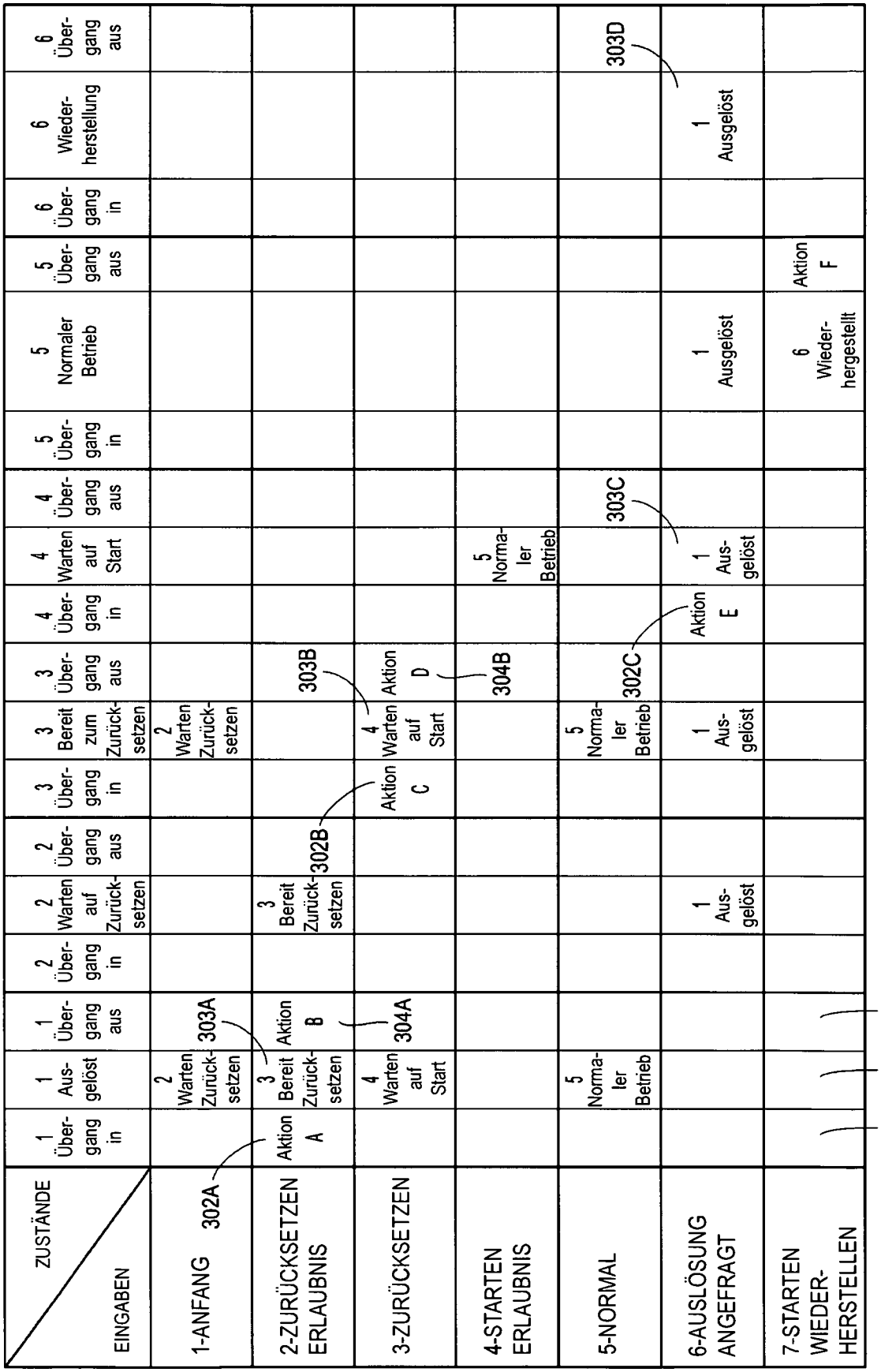

300
ZUSTÄNDE
EINGABEN
1 Über-gang in
1 Aus-gelöst
1 Über-gang aus
2 Über-gang in
2 Warten auf Zurück-setzen
2 Über-gang aus
3 Über-gang in
3 Bereit zum Zurück-setzen
3 Über-gang aus
4 Über-gang in
4 Warten auf Start
4 Über-gang aus
5 Über-gang in
5 Normaler Betrieb
5 Über-gang aus
6 Über-gang in
6 Wieder-herstellung
6 Über-gang aus
1-ANFANG
2-ZURÜCKSETZEN ERLAUBNIS
3-ZURÜCKSETZEN
4-STARTEN ERLAUBNIS
5-NORMAL
6-AUSLÖSUNG ANGEFRAGT
7-STARTEN WIEDER-HERSTELLEN
302A
303A
303B
302B
304A
304B
302C
303C
303D
304E
303E
302E
Aktion A
Aktion B
Aktion C
Aktion D
Aktion E
Aktion F
2 Warten Zurück-setzen
3 Bereit Zurück-setzen
4 Warten auf Start
5 Norma-ler Betrieb
1 Aus-gelöst
6 Wieder-hergestellt
1 Ausgelöst

FIG. 6

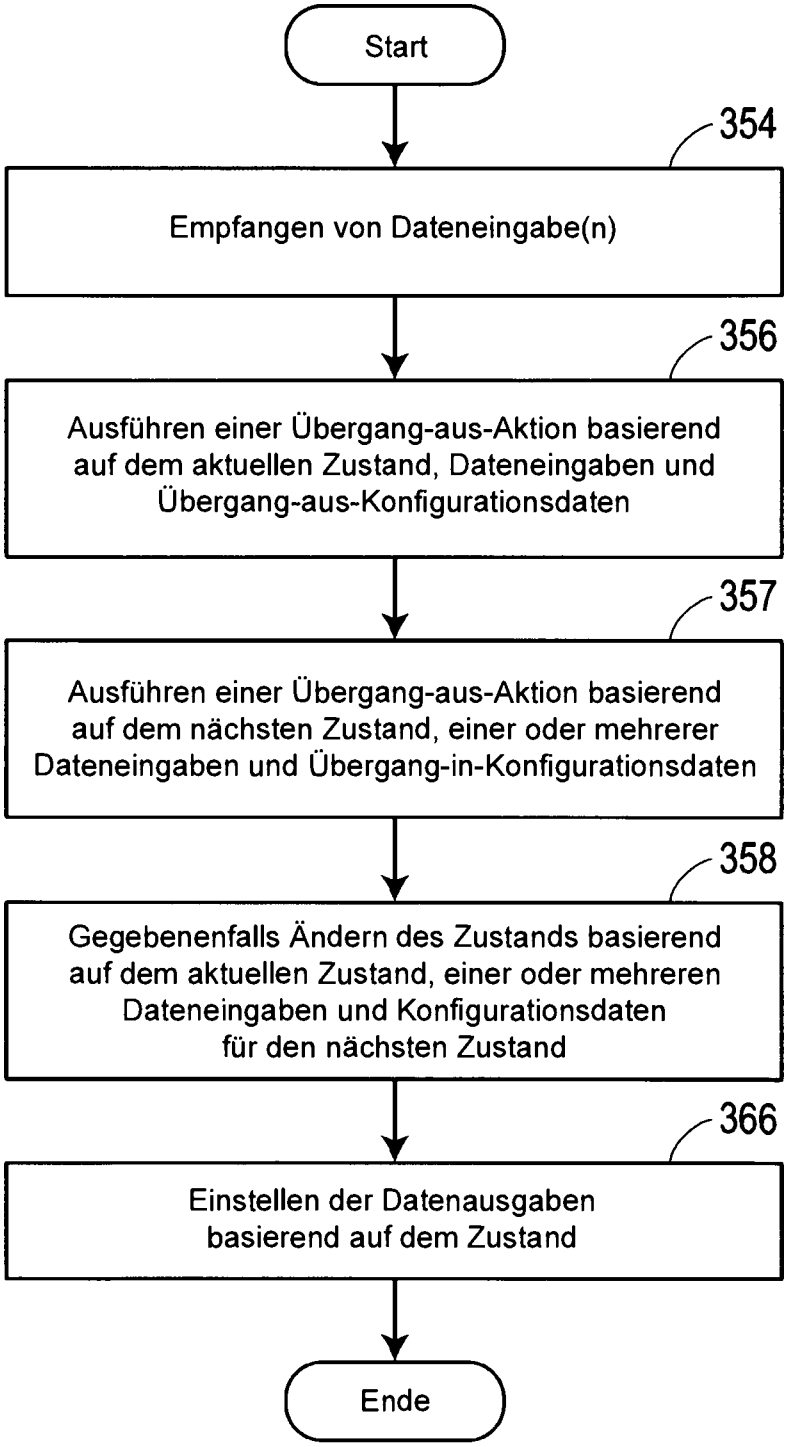

Start
354
Empfangen von Dateneingabe(n)
356
Ausführen einer Übergang-aus-Aktion basierend auf dem aktuellen Zustand, Dateneingaben und Übergang-aus-Konfigurationsdaten
357
Ausführen einer Übergang-aus-Aktion basierend auf dem nächsten Zustand, einer oder mehrerer Dateneingaben und Übergang-in-Konfigurationsdaten
358
Gegebenenfalls Ändern des Zustands basierend auf dem aktuellen Zustand, einer oder mehreren Dateneingaben und Konfigurationsdaten für den nächsten Zustand
366
Einstellen der Datenausgaben basierend auf dem Zustand
Ende

FIG. 7

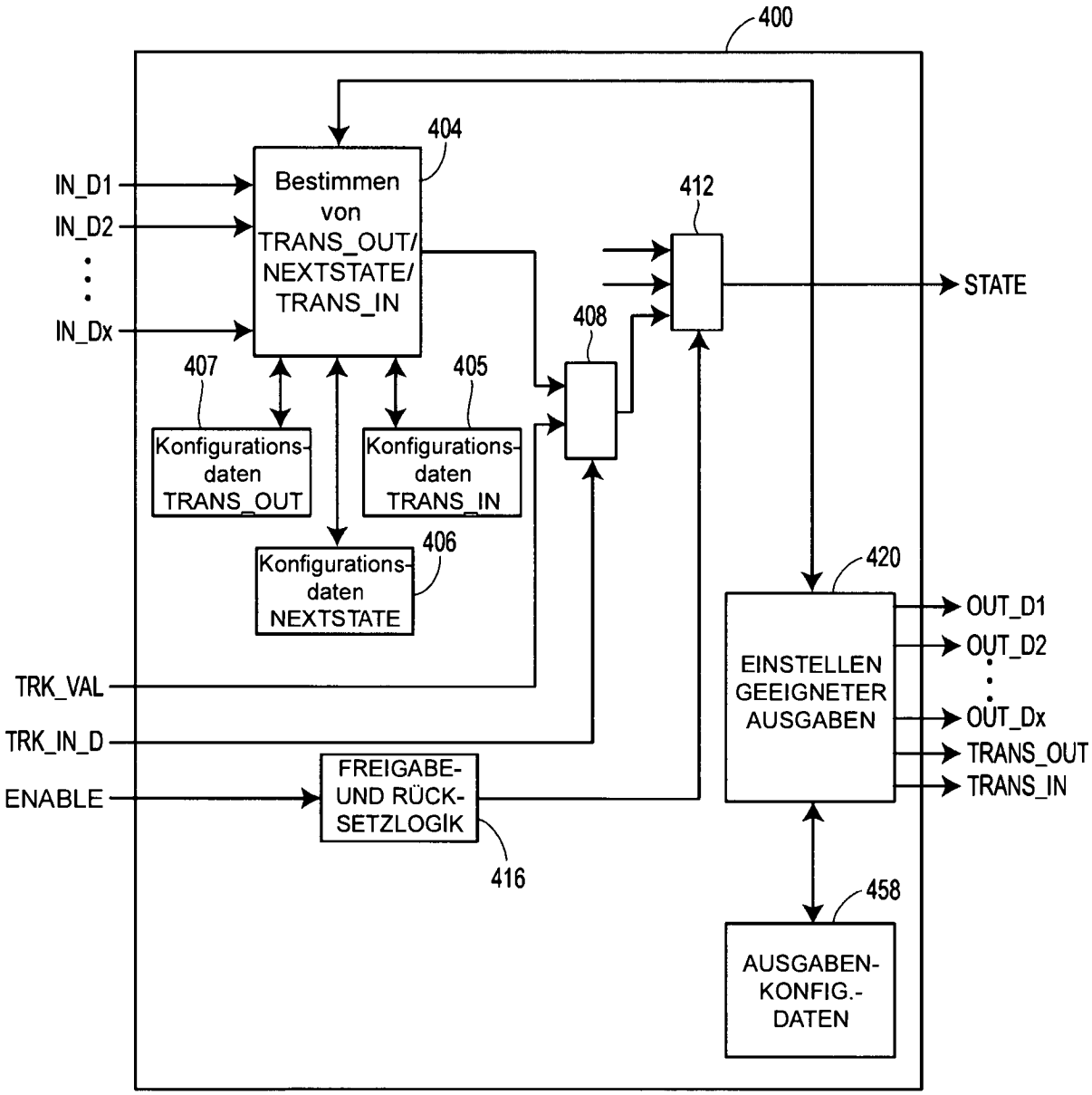

400
404
Bestimmen von TRANS_OUT/ NEXTSTATE/ TRANS_IN
IN_D1
IN_D2
IN_Dx
412
STATE
408
407
405
Konfigurations-daten TRANS_OUT
Konfigurations-daten TRANS_IN
406
Konfigurations-daten NEXTSTATE
420
EINSTELLEN GEEIGNETER AUSGABEN
OUT_D1
OUT_D2
OUT_Dx
TRANS_OUT
TRANS_IN
TRK_VAL
TRK_IN_D
ENABLE
FREIGABE- UND RÜCK-SETZLOGIK
416
458
AUSGABEN-KONFIG.-DATEN

FIG. 8

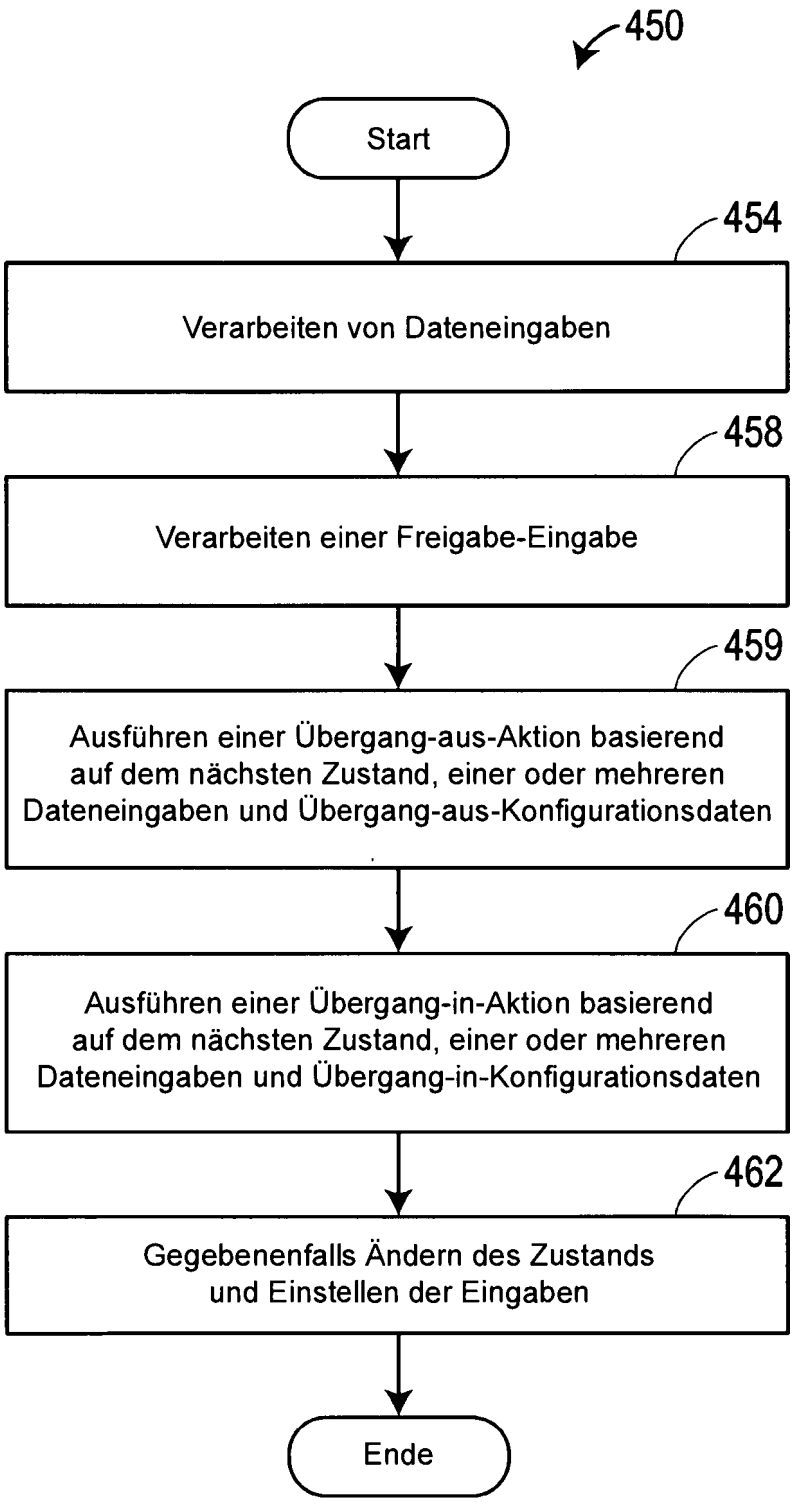

450
Start
454
Verarbeiten von Dateneingaben
458
Verarbeiten einer Freigabe-Eingabe
459
Ausführen einer Übergang-aus-Aktion basierend auf dem nächsten Zustand, einer oder mehreren Dateneingaben und Übergang-aus-Konfigurationsdaten
460
Ausführen einer Übergang-in-Aktion basierend auf dem nächsten Zustand, einer oder mehreren Dateneingaben und Übergang-in-Konfigurationsdaten
462
Gegebenenfalls Ändern des Zustands und Einstellen der Eingaben
Ende

FIG. 9

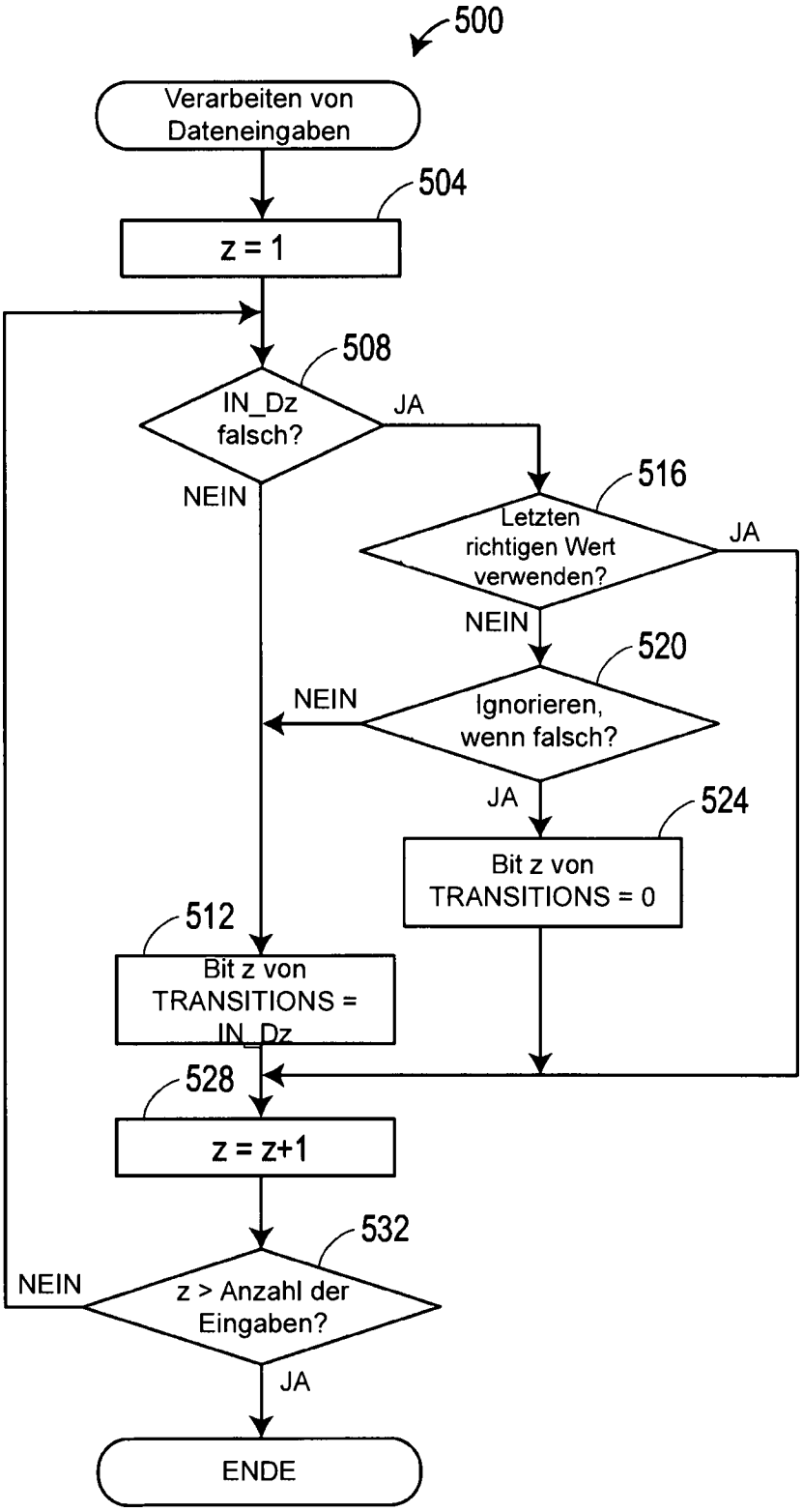

500
Verarbeiten von Dateneingaben
504
z = 1
508
IN_Dz falsch?
JA
NEIN
516
Letzten richtigen Wert verwenden?
JA
NEIN
520
Ignorieren, wenn falsch?
NEIN
JA
524
Bit z von TRANSITIONS = 0
512
Bit z von TRANSITIONS = IN_Dz
528
z = z+1
532
z > Anzahl der Eingaben?
NEIN
JA
ENDE

FIG. 10

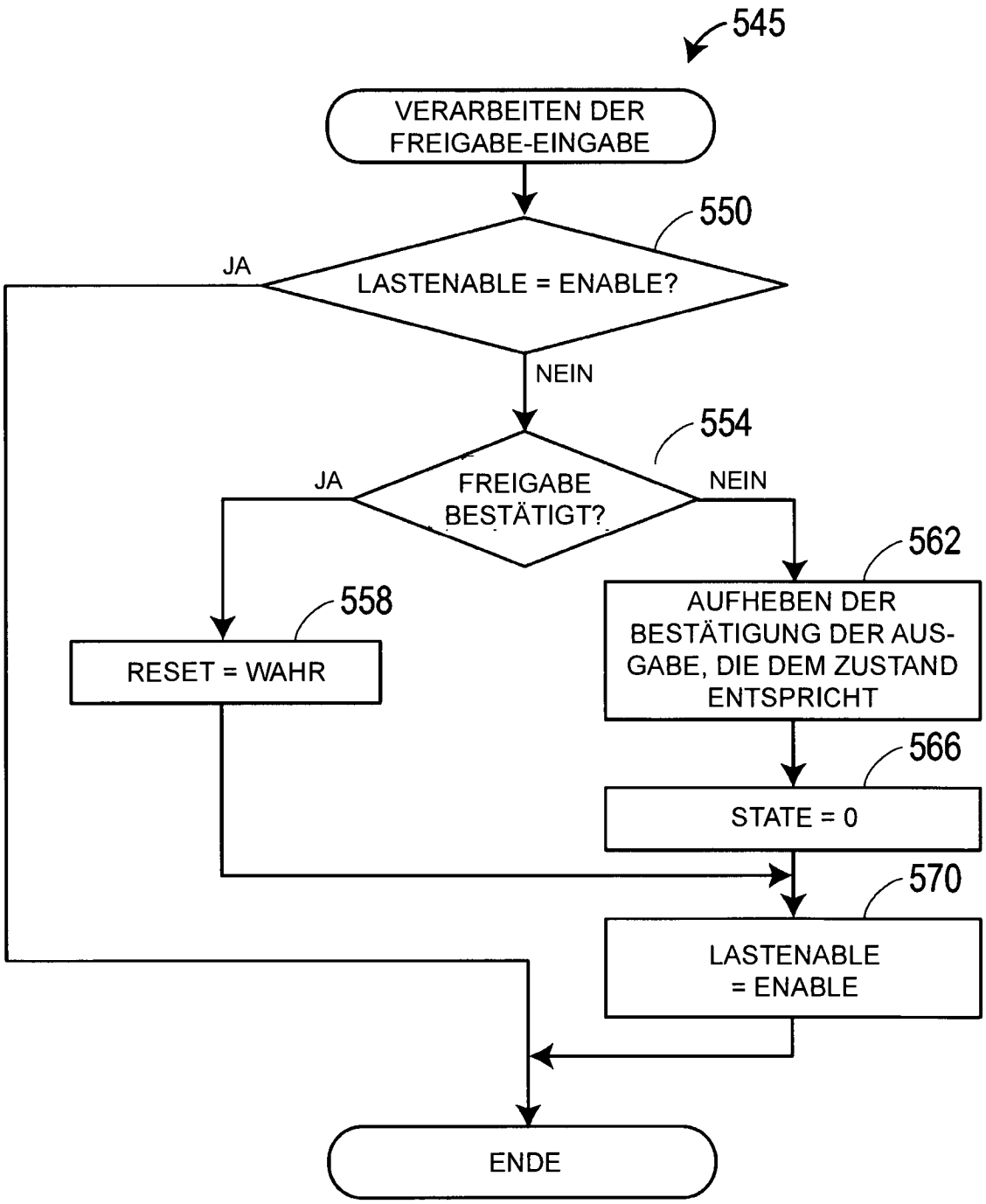

545
VERARBEITEN DER FREIGABE-EINGABE
550
LASTENABLE = ENABLE?
JA
NEIN
554
FREIGABE BESTÄTIGT?
JA
NEIN
558
RESET = WAHR
562
AUFHEBEN DER BESTÄTIGUNG DER AUSGABE, DIE DEM ZUSTAND ENTSPRICHT
566
STATE = 0
570
LASTENABLE = ENABLE
ENDE

FIG. 11

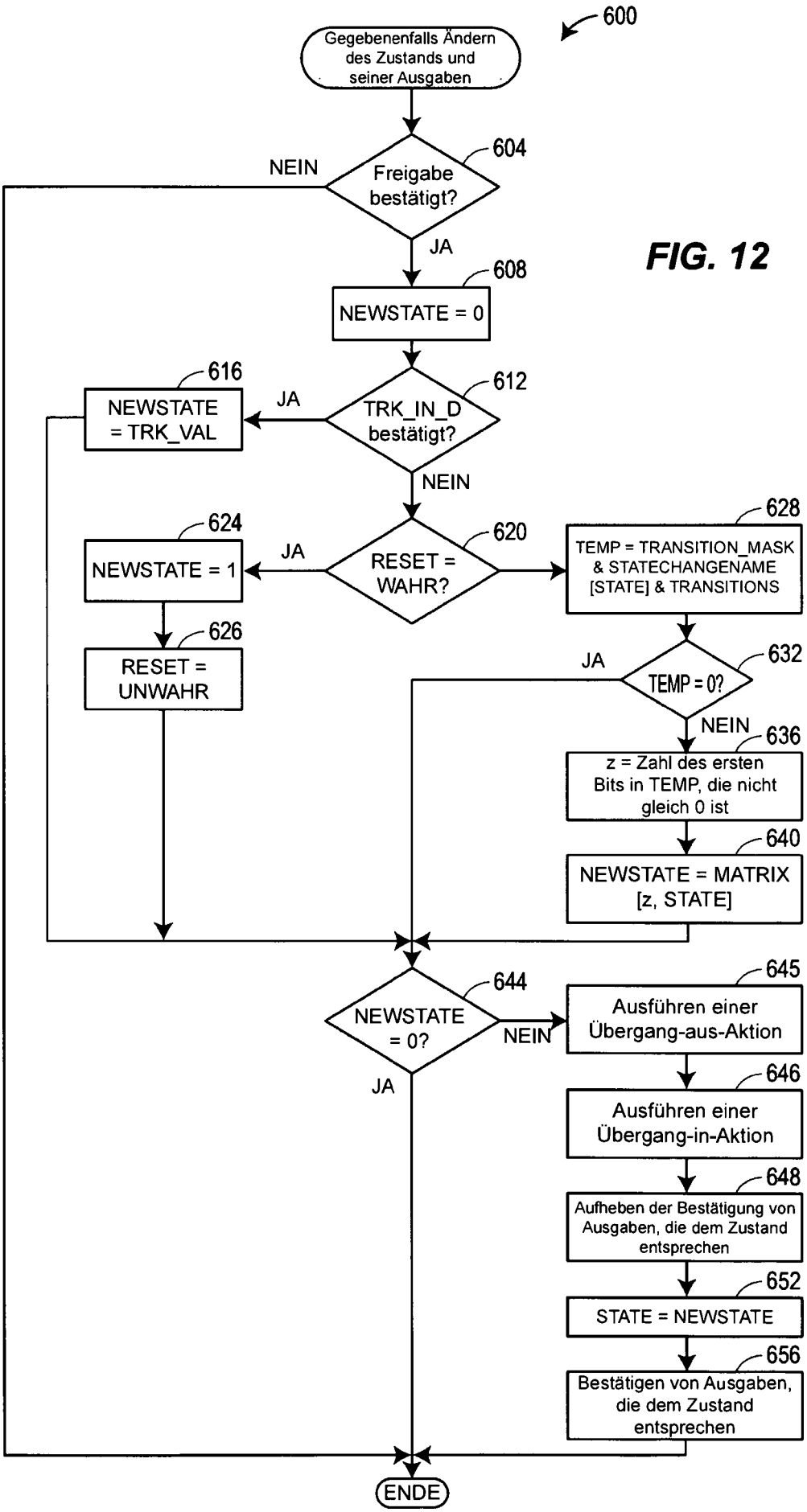
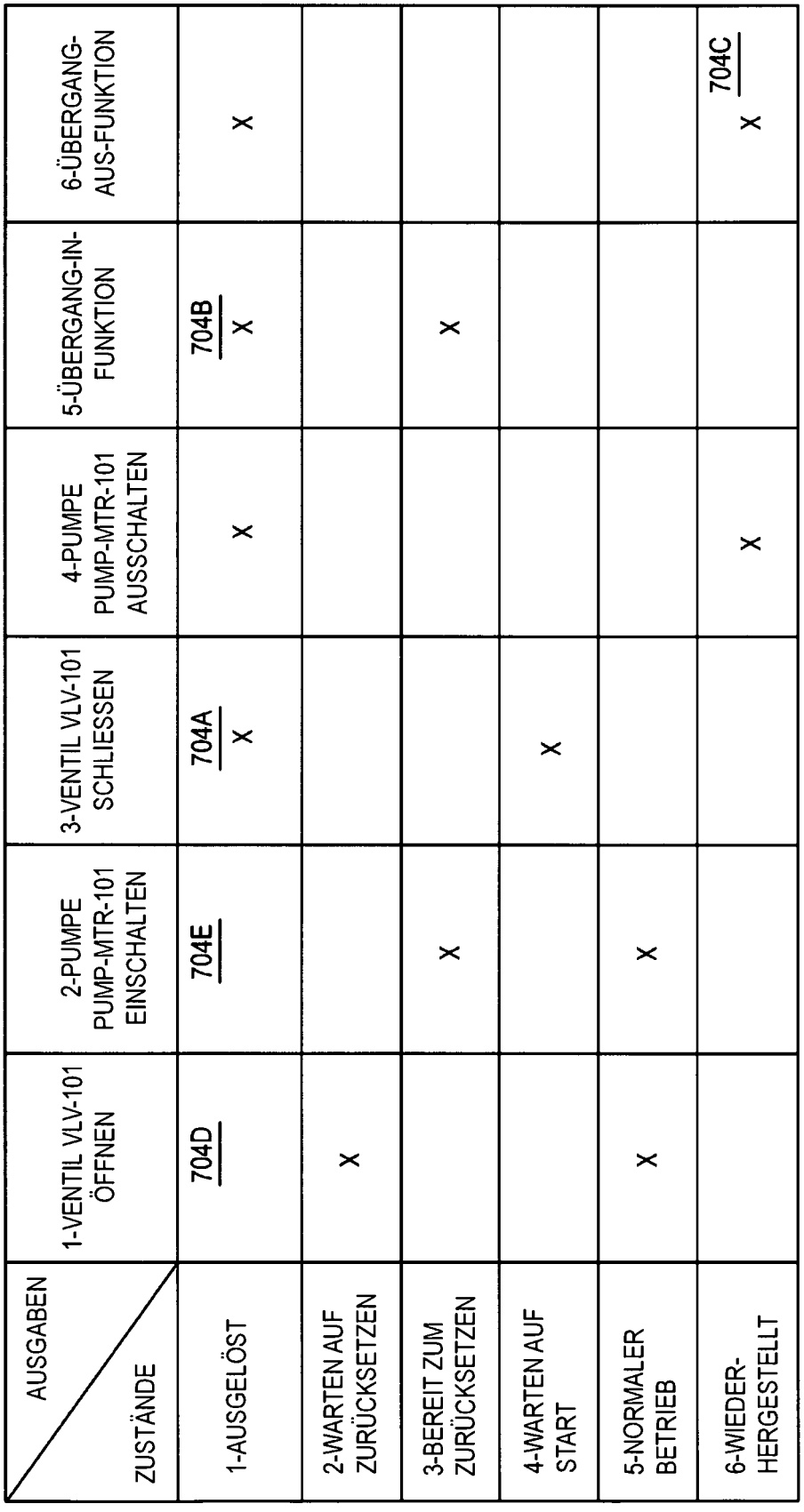

600
Gegebenenfalls Ändern des Zustands und seiner Ausgaben
604
Freigabe bestätigt?
NEIN
JA
608
NEWSTATE = 0
612
TRK_IN_D bestätigt?
616
NEWSTATE = TRK_VAL
JA
NEIN
620
RESET = WAHR?
624
NEWSTATE = 1
JA
626
RESET = UNWAHR
628
TEMP = TRANSITION_MASK & STATECHANGENAME [STATE] & TRANSITIONS
632
TEMP = 0?
JA
NEIN
636
z = Zahl des ersten Bits in TEMP, die nicht gleich 0 ist
640
NEWSTATE = MATRIX [z, STATE]
644
NEWSTATE = 0?
NEIN
JA
645
Ausführen einer Übergang-aus-Aktion
646
Ausführen einer Übergang-in-Aktion
648
Aufheben der Bestätigung von Ausgaben, die dem Zustand entsprechen
652
STATE = NEWSTATE
656
Bestätigen von Ausgaben, die dem Zustand entsprechen
ENDE

FIG. 12



AUSGABEN / ZUSTÄNDE	1-VENTIL VLV-101 ÖFFNEN	2-PUMPE PUMP-MTR-101 EINSCHALTEN	3-VENTIL VLV-101 SCHLIESSEN	4-PUMPE PUMP-MTR-101 AUSSCHALTEN	5-ÜBERGANG-IN-FUNKTION	6-ÜBERGANG-AUS-FUNKTION
1-AUSGELÖST	704D	704E	704A X	X	704B X	X
2-WARTEN AUF ZURÜCKSETZEN	X					
3-BEREIT ZUM ZURÜCKSETZEN		X			X	
4-WARTEN AUF START			X			
5-NORMALER BETRIEB	X	X				
6-WIEDER-HERGESTELLT				X		704C X

FIG. 13

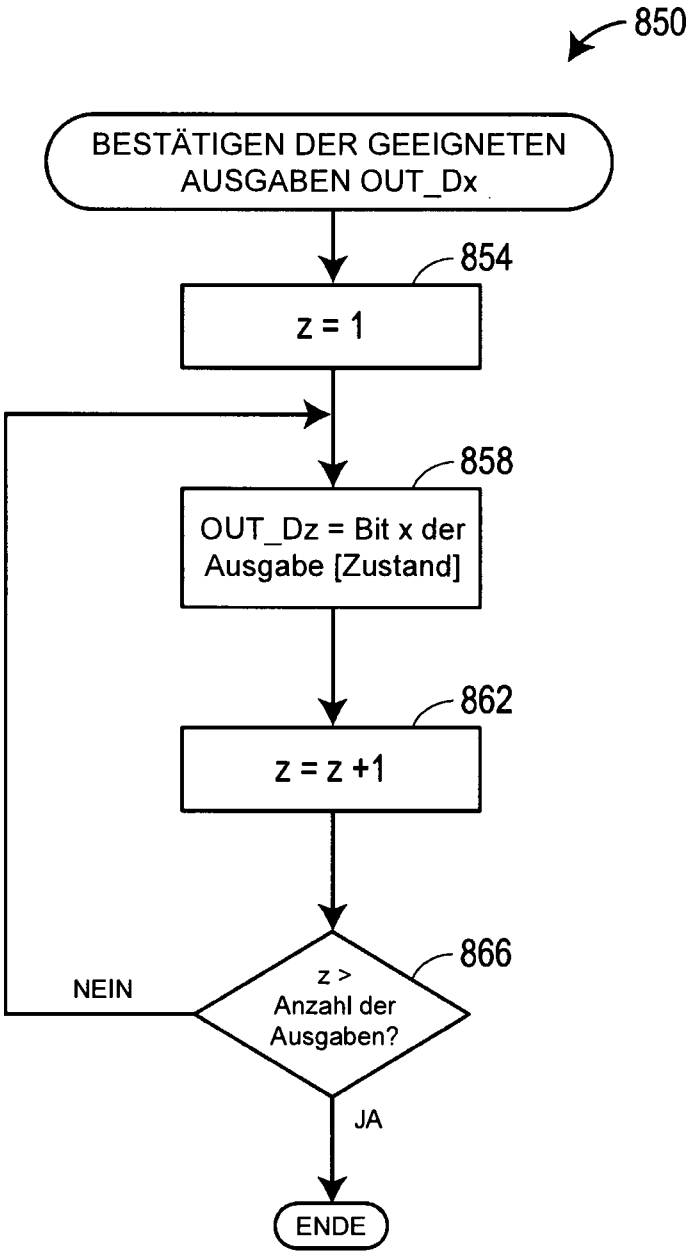

850
BESTÄTIGEN DER GEEIGNETEN AUSGABEN OUT_Dx
854
z = 1
858
OUT_Dz = Bit x der Ausgabe [Zustand]
862
z = z +1
866
z > Anzahl der Ausgaben?
NEIN
JA
ENDE

FIG. 14

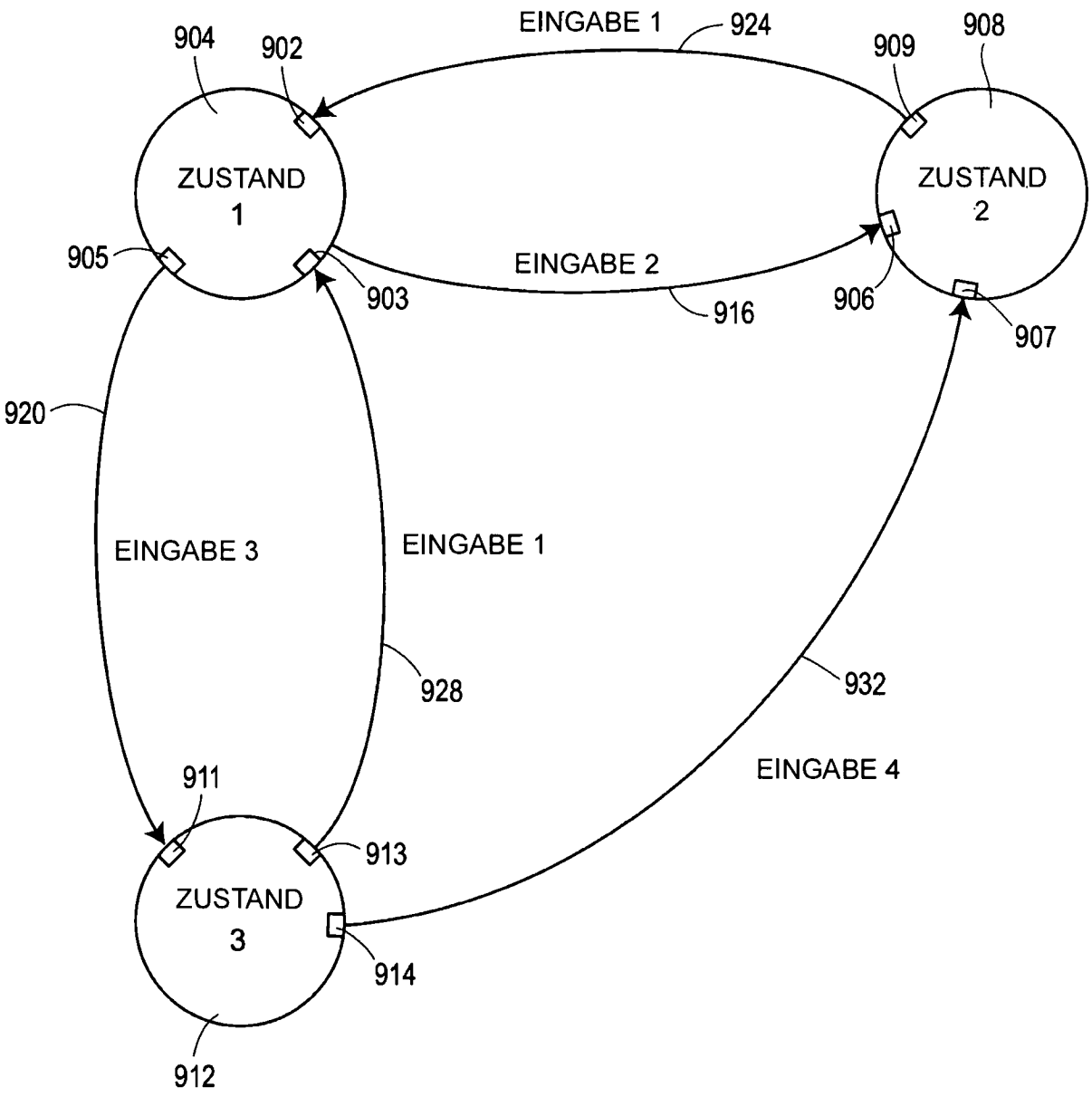

EINGABE 1
924
904
902
909
908
ZUSTAND 1
ZUSTAND 2
905
EINGABE 2
903
916
906
907
920
EINGABE 3
EINGABE 1
928
932
EINGABE 4
911
913
ZUSTAND 3
914
912

FIG. 15