(19) österreichisches patentamt

(10) **AT 12 998 U1 2013-03-15**

(12) **Gebrauchsmusterschrift**

(21) Anmeldenummer: GM 8/2012
(22) Anmeldetag: 12.01.2012
(24) Beginn der Schutzdauer: 15.01.2013
(45) Veröffentlicht am: 15.03.2013

(51) Int. Cl. : ***G05B 19/05*** (2006.01)
G06F 11/16 (2006.01)

(56) Entgegenhaltungen:
US 2008125886 A1
DE 102009050449 B3

(73) Gebrauchsmusterinhaber:
BACHMANN GMBH
6800 FELDKIRCH (AT)

(72) Erfinder:
MARX SÖNKE
FELDKIRCH (AT)
DORNER MARCEL
ALBERSCHWENDE (AT)
KHÜNY GERHARD
NENZING (AT)
BERTEL WOLFGANG
FELDKIRCH (AT)

(54) **REDUNDANTES STEUERUNGSSYSTEM SOWIE STEUERRECHNER UND PERIPHERIEEINHEIT**

(57) Redundantes Steuerungssystem mit zumindest einem ersten Steuerrechner (1) und einem zweiten Steuerrechner (2), die miteinander über eine Redunanzkopplung (3, 4) verbunden sind, und zumindest einer der Steuerrechner (1, 2) zur Ausführung von Steuerungsaufgaben mit zumindest einer Peripherie-Einheit (7, 8, 9) über zumindest ein Bussystem (5, 6) Daten austauscht, wobei auf den mindestens zwei Steuerrechnern (1, 2) jeweils mehrere unterschiedliche Applikationen (11,12,13; 11', 12', 13') gleichzeitig und unabhängig voneinander ablaufen, dass die jeweilige Applikation (11, 12, 13) auf dem einen Steuerrechner (1) in identischer Form als gespiegelte Applikation (11', 12', 13') auf dem anderen Steuerrechner (2) abläuft und im Fehlerfall die von der fehlerhaften Applikation (11, 12, 13) angesteuerte Peripherie-Einheit (7-9) auf den anderen Steuerrechner und die dort noch lauffähige gleiche Applikation (11', 12', 13') umschaltet

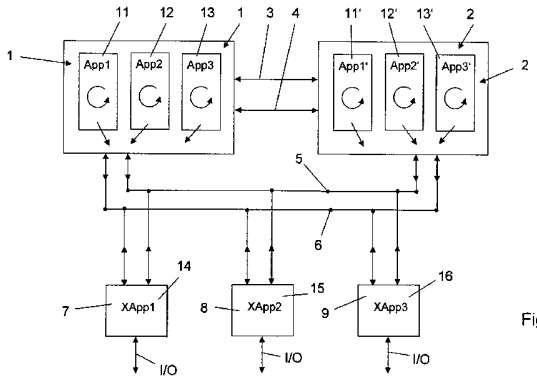


Fig. 2

AT 12 998 U1 2013-03-15

Beschreibung

REDUNDANTES STEUERUNGSSYSTEM SOWIE STEUERRECHNER UND PERIPHERIE-EINHEIT

[0001] Gegenstand der Erfindung ist ein redundantes Steuersystem sowie ein Steuerrechner und eine Peripherieeinheit nach dem Oberbegriff des Anspruches 1.

[0002] Ein derartiges redundantes Steuerungssystem ist beispielsweise mit dem Gegenstand der DE 100 30 329 C1 bekannt geworden.

[0003] Bei dieser bekannten Einrichtung sind zwei Steuerrechner 1 und 2 über ein als Redundanzkopplung ausgestaltetes erstes Bussystem miteinander verbunden und tauschen Daten miteinander aus, während jeder der beiden Steuerrechner zur Ausführung von Steuerungsaufgaben über ein weiteres Bussystem mit einer oder mehreren Peripherieeinheiten verbunden ist. Dieses redundante Steuerungssystem soll sicherstellen, dass die Peripherie weiter mit Daten versorgt wird, auch wenn einer der beiden Steuerrechner ausfällt. Bei Ausfall des einen Steuerrechners wird über die Redundanzkopplung auf den zweiten Steuerrechner umgeschaltet, sodass die Peripherie weiter mit Daten versorgt wird.

[0004] Zur Sicherstellung des Funktionszustandes der beiden Steuerrechner senden beide Steuerrechner über das Bussystem an die Peripherieeinheit jeweils zyklisch wechselnde Lebenszeichen. Die Peripherieeinheit ist dazu ausgebildet, zu überprüfen, ob innerhalb einer vorgebbaren Zeitdauer ein Wechsel eines Lebenszeichens aufgetreten ist. Bei Ausbleiben eines Wechsels wird ein Fehler des jeweiligen Steuerrechners erkannt und auf den anderen Steuerrechner zur Ausführung der Steuerungsaufgabe umgeschaltet.

[0005] Nachteil dieser Anordnung ist, dass die Erzeugung von zyklisch abwechselnden Lebenszeichen mit einem hohen Steuerungsaufwand verbunden ist. Es kann im Übrigen zu einem unerwünschten Datenverlust kommen, weil erst bei der Auswertung der abwechselnd eintreffenden Lebenszeichen von einer Peripherie-Einheit entschieden wird, ob ein Defekt vorliegt oder nicht. Während dieser Entscheidungszeit können jedoch Daten verloren gehen, weil noch nicht umgeschaltet wurde.

[0006] Allerdings läuft bei der DE 100 30 329 C1 auf jedem Steuerrechner nur eine einzige eine Applikation. Hierunter wird nicht ein Multi-Thread-Betrieb verstanden, denn ein klassischer Multi-Thread-Betrieb hat unterschiedliche Tasks mit unterschiedlichen Zykluszeiten in einem einzigen Anwenderprogramm.

[0007] Wenn dieses einzige Anwenderprogramm gestoppt wird, werden auch alle Tasks im Rahmen des Multi-Thread-Task gestoppt oder modifiziert oder ausgetauscht, worauf dann auf jeden Fall die komplette Applikation steht und nicht mehr weiterlaufen kann.

[0008] Erst in diesem Versagensfall wird das permanent ausgetauschte Lebenszeichen als ausgefallen angesehen und auf den anderen redundanten Steuerrechner umgeschaltet.

[0009] Nachteil der bekannten Lösung ist demgemäß, dass wenn das jeweilig auf den beiden Steuerrechnern identisch ablaufende Hautprogramm (Applikation) gestoppt oder modifiziert wird, erlöschen somit auch alle anderen untergeordneten Tasks und die Peripherie wird nicht mehr mit Daten versorgt.

[0010] Der Erfindung liegt deshalb die Aufgabe zu Grunde ein redundantes Steuerungssystem sowie einen Steuerrechner und eine daran angeschlossene Peripherieeinheit so weiter zu bilden, dass im Fehlerfall ohne Datenverlust von einem Steuerrechner auf den anderen umgeschaltet werden kann und dass die aufwendige Erzeugung von Lebenszeichen entfallen kann, was - wie oben dargestellt - mit höheren Laufzeiten und mit einer verminderten Zuverlässigkeit verbunden ist.

[0011] Zur Lösung der gestellten Aufgabe ist die Erfindung durch die technischen Lehren der unabhängigen Ansprüche gemäß den nachfolgenden Abschnitten I bis V gekennzeichnet. Die

jeweils geschilderten Steuerrechner der nachfolgenden Abschnitte I bis V sind in der Lage, bei einem Fehler in einem der Steuerrechner auf den anderen Steuerrechner umzuschalten und ohne Datenverlust, die Daten von dem anderen (redundanten) Steuerrechner abzurufen und der Peripherie zu zuführen.

I. REDUNDANZ DURCH AUF DEN STEUERRECHNERN GETRENNT VONEINANDER ABLAUFENDE, GEKLONTE APPLIKATIONEN MIT GEGEBENENFALLS UNTERSCHIEDLICHEN ZYKLUSZEITEN.

[0012] Der Erfindung bildet deshalb ein aus der DE 100 30 329 C1 bekanntes, redundantes Steuerungssystem sowie einen Steuerrechner und eine daran angeschlossene Peripherieeinheit der eingangs genannten Art insofern weiter, dass bei einem Fehlereintritt in einem der beiden Steuerrechner die Peripherie nicht vollständig von den Steuerrechnern abgeschaltet wird, wenn die auf dem jeweiligen Steuerrechner ablaufende Applikation zum Stillstand kommt.

[0013] Die Erfindung sieht deshalb vor, auf einem redundanten Steuerungssystem mit mindestens zwei Steuerrechnern mehrere Applikationen ggf. mit mehreren Tasks gleichzeitig und unabhängig voneinander ablaufen zu lassen, wobei die Applikationen (Tasks) durchaus unterschiedliche Zykluszeiten haben können.

[0014] Wesentlich ist demnach, dass auf den mindestens zwei Steuerrechnern jeweils mehrere unterschiedliche Applikationen gleichzeitig und unabhängig voneinander ablaufen, dass die jeweilige Applikation auf dem einen Steuerrechner in identischer Form als gespiegelte Applikation auf dem anderen Steuerrechner abläuft und im Fehlerfall die von der fehlerhaften Applikation angesteuerte Peripherie-Einheit auf den anderen Steuerrechner und die dort noch lauffähige gleiche Applikation umschaltet. Es können deshalb bei der Umschaltung keine Daten verloren gehen, weil die auf dem anderen Steuerrechner ablaufende Applikation mit der gleichen Zykluszeit und den gleichen Werten zum gleichen Zeitpunkt wie die als fehlerhaft erkannte Spiegel-Applikation weiter arbeitet. Die Peripherie-Einheit merkt also keine Umschaltung und arbeitet mit den gleichen Daten, die auf der Ursprungsapplikation vorhanden waren mit den identischen Daten auf der Spiegel-Applikation weiter.

[0015] Das System nach der Erfindung sieht deshalb eine redundante Steuerung (RPLC-redundant programable logic controller) von mindestens zwei Steuerrechnern einschliesslich einer zwischen den Steuerrechnern geschalteten redundanten Kopplung vor, die z.B. als Netzwerkverbindung (Redundancy Interconnection = RI) ausgebildet ist, und einem redundanten IO-Network (= redundante Busleitungen zwischen mehreren Peripheriegeräten) vor. Hierbei kann vorgesehen sein, dass die redundanten Steuerrechner über ein Fabriknetzwerk (Factory Network) mit dort angeordneten Benutzerschnittstellen angesteuert werden. Hierauf ist die Erfindung jedoch nicht beschränkt. Das mit den Benutzerschnittstellen ausgerüstete Factory Network kann auch entfallen Das Factory Network hat jedoch nichts mit den Merkmalen der zueinander redundanten Steuerrechner zu tun.

[0016] Die beiden redundanten Steuerrechner (RPLCs) kommunizieren miteinander über die Redundancy Interconnection (Verbindungsbus). Die Steuerrechner tauschen Werte und auch Programme über diese Schnittstelle aus und synchronisieren sich ständig. Deshalb werden die auf dem einen Steuerrechner ablaufenden Applikationen genau identisch und synchron in Bezug zu den auf dem anderen Steuerrechner ablaufenden Applikationen gehalten. Für die Kommunikation zwischen den beiden Steuerrechnern (RPLCs) kann auch das IO-Netzwerk verwendet werden. Über das IO-Netzwerk kommunizieren die RPLCs mit den Peripherie-Einheiten (IO-Devices). Die Peripherie-Einheiten (IO-Devices) sind als Slaves ausgebildet. Es handelt sich beispielsweise um Maschinensteuerungen, bei denen der Wert eines oder mehrerer Parameter z.B. für eine Pumpensteuerung, eine Logikeinheit, eine Leuchteinheit und dgl. gesetzt werden.

[0017] Die Topologie des IO-Networks kann von Anwendung zu Anwendung unterschiedlich sein. Dieses bedeutet, dass auch ein einfaches Netzwerk, wie z.B. ein serielles Netzwerk, funktioniert. Über das Factory Netzwerk können die redundanten Steuerrechner (RPLCs) mit den Benutzerschnittstellen (SolutionCenter, SCADA, Visualisierung) verbunden sein. Die

Busstruktur des IO-Networks kann jedoch auch als Ethernet, Profibus, CAN-Bus, Feldbus und dgl. ausgebildet sein.

[0018] Wesentliches Merkmal der Erfindung ist demnach, dass in jeweils einem Steuerrechner mehrere, voneinander unabhängige Applikationen (Anwenderprogramme) gleichzeitig und parallel ablaufen, die später auch als redundante PLC-Projekte (RPLCs) bezeichnet werden.

[0019] Mit der erfindungsgemäßen Steuerung ist nunmehr möglich, mehrere redundante PLC- (programable logical controller auch SPS- Steuerungen genannt) Projekte (Applikationen) mit unterschiedlichen Zykluszeiten und unterschiedlichen Prioritäten laufen zu lassen. Jede dieser Applikationen (PLC-Projekte) hat einen eigenen Namen und kann so modifiziert, gelöscht oder installiert werden, ohne dass die anderen Applikationen (PLC-Projekte) beeinflusst werden. Die Synchronisation der redundanten Applikationen geschieht im Zyklus der jeweiligen Applikation. Dadurch ist gewährleistet, dass die redundanten Applikationen, die synchron auf den beiden Steuerrechnern zur gleichen Zeit ablaufen, die gleichen Startbedingungen haben.

[0020] Bisher war es im Stand der Technik nicht bekannt, solche Applikationen (also z.B. SPS - speicher-programmierbare Steuerungen) eigenständig und unabhängig nebeneinander laufend auf einem einzigen Steuerrechner anzuordnen und auf dem zweiten Steuerrechner die identischen (geklonten) Applikationen (PLC-Projekte) zur gleichen Zeit in gleicher Weise ablaufen zu lassen.

[0021] Vorteil der oben genannten Maßnahme ist, dass die Peripherie weiter lebt und einen Datenaustausch mit den Steuerrechnern erfährt, auch wenn einzelne redundante Applikationen im Steuerrechner zum Erliegen kommen.

[0022] Vorteil der Erfindung ist, dass im Vergleich zum Stand der Technik nach der DE 100 30 329 C1 nicht nur eine einzige Applikation jeweils auf einem Steuerrechner abläuft, sondern dass erfindungsgemäß eine Anzahl von unabhängig voneinander ablaufenden Applikationen in jeweils einem Steuerrechner ablaufen und wobei diese Applikationen nicht voneinander abhängig sind.

[0023] Stürzt eine Applikation aus irgendwelchen Gründen auf dem einen Steuerrechner ab, läuft die redundante andere Applikation auf dem anderen Steuerrechner weiter und versorgt die mit den Daten dieser Applikation arbeitende Peripherie-Einheit ohne Unterbrechung mit den identischen Daten weiter.

[0024] Der Begriff „redundante Applikation" meint, dass die Applikation sowohl auf dem Steuerrechner 1 als auch auf dem Steuerrechner 2 identisch abläuft und identische Programmschleifen und die gleichen Zeitbedingungen erfüllen, so dass der Steuerrechner 1 ein Klon des Steuerrechners 2 darstellt und auf jedem Steuerrechner eine identische Anzahl von identischen Applikationen ablaufen.

[0025] Wichtig hierbei ist, dass beim Versagen der einen Applikation auf einem Steuerrechner die gleiche Applikation auf dem anderen Steuerrechner weiterläuft, und über die zwischen den beiden Steuerrechnern angeordnete Busleitung (Redundancy Interconnection) erfährt der andere Steuerrechner, dass die auf dem ersten Steuerrechner ablaufende (Zwillings-) Applikation zum Erliegen gekommen ist. Die zweite Steuerung schreibt ihr Werte auch auf den Bus (IO-Network) zur Peripherie. Jedoch entscheidet die Peripherie, ob sie die Werte von dem Steuerrechner 1 oder 2 nimmt. Damit gelingt es die von der Applikation abhängige Peripherie-Einheit unterbrechungsfrei mit den Daten der Zwillingsapplikation weiter zu versorgen.

[0026] Im übertragenen Sinn gesprochen handelt es sich bei den Steuerrechnern 1 und 2 um zueinander geklonte Steuerrechner, auf denen voneinander unabhängige identische Applikationen ablaufen. Die Erfindung ist nicht auf eine einzige, auf einem Steuerrechner ablaufende Applikation beschränkt, vielmehr können mehrere, auch unterschiedliche Applikationen, die jedoch vollkommen unabhängig voneinander sind, auf den Steuerrechnern ablaufen.

[0027] Für den Fall, dass einer den Steuerrechner gewartet oder die Peripherieeinheiten gewartet werden müssen, besteht damit der Vorteil, dass die spezifischen Applikationen, die auf

den Steuerrechnern synchron ablaufen und die eine bestimmte Peripherie bedienen, gestoppt werden können oder modifiziert werden können, wobei die andere Peripherie durch die anderen Applikationen weiterhin am Leben bleibt.

[0028] Dies bedeutet, dass die Steuerrechner zwischen sich umschalten und wenn eine bestimmte Applikation auf dem einen Steuerrechner zum Erliegen gekommen ist, läuft dann die gleiche Applikation auf dem anderen Steuerrechner weiter. Die zweite Steuerung schreibt ihre Werte auch auf den Bus (IO-Network) zur Peripherie. Jedoch entscheidet die Peripherie, ob sie die Werte von dem Steuerrechner 1 oder 2 nimmt.

[0029] Wesentlich ist also, dass unterschiedliche, jedoch zueinander geklonte Applikationen auf den Rechnern laufen, die unabhängig voneinander sind und der Anwender in der Lage ist, die jeweiligen Applikationen einzeln zu behandeln ohne die anderen Applikationen zu stören.

[0030] Der Anwender kann demnach individuell jede einzelne der autark ablaufenden Applikationen unterschiedlich aufteilen, modifizieren, stoppen, starten, oder löschen, ohne dass die anderen Applikationen davon betroffen sind. Damit wird gewährleistet, dass die Peripherie, die durch diese Applikation weiter mit Daten bedient wird, und nicht zum Stillstand kommt. Ein Verlust von Daten beim Umschalten vom einen auf den anderen Steuerrechner ist deshalb - im Gegensatz zum Stand der Technik - nicht mehr möglich.

[0031] Erfindungsgemäß ist vorgesehen, dass jede von den auf den einzelnen Steuerrechnern ablaufenden, voneinander unabhängigen Applikationen Multitask- und auch Multi-Thread- fähig ist. Das bedeutet, dass die Applikationen aus verschiedenen Unterprogrammen bestehen können und die Unterprogramme vom Status des Hauptprogramms oder Anwenderprogramms abhängig sind.

[0032] Wichtig ist jedoch, dass jede Applikation vollkommen unabhängig (autark) von der anderen, parallel im Steuerrechner ablaufenden Applikation abläuft. Die Applikationen sind demnach als selbstständige, allein stehende Applikationen geschrieben, die nicht auf einen Datenaustausch mit einer parallel dazu ablaufenden und unabhängig davon bestehenden weiteren Applikation angewiesen sind.

[0033] Die unabhängig voneinander ablaufenden und sich gegenseitig nicht beeinflussenden Applikationen laufen jeweils in einem gemeinsamen Mikroprozessor ab. Es ist jedoch in einer Weiterbildung der Erfindung auch vorgesehen, dass der Steuerrechner als Multi-Core-Prozessor ausgebildet ist auf dessen unterschiedlichen Cores die Applikationen laufen.

[0034] In einer anderen Ausgestaltung ist es vorgesehen, dass die unterschiedlichen und unabhängig voneinander ablaufenden Applikationen sich unterschiedliche, von einander unabhängige Speicherbereiche im Hauptsspeicher teilen oder auch komplette andere Speicher benutzen, um zu gewährleisten, dass beim Fehler eines Speichers nicht alle Applikationen betroffen sind.

[0035] Es muss jegliche Einflussnahme der einen Applikation auf die andere ausgeschlossen werden, sodass beim Absturz der einen Applikation die andere Applikation nicht in Gefahr gerät.

[0036] Wenn aber eine Applikation stillsteht oder fehlerbedingt seine Arbeit einstellt, beeinflusst dies die anderen Applikationen nicht. Diese laufen im Speicherbereich weiter.

[0037] Wichtig ist also, dass sich eine Anzahl von parallelen, unabhängig voneinander ablaufenden Applikationen einen einzigen Prozessor teilen, worauf die Erfindung jedoch nicht beschränkt ist. Wie vorhin ausgeführt, können diese Applikationen auch auf verschiedenen Prozessoren ablaufen, die jeweils alle in einem Steuerrechner zusammengefasst sind.

[0038] Es ist im Stand der Technik bekannt, einen Dual-und/ oder Quad-Core-Prozessor zu verwenden, von den sich hieraus ergebenden Vorteilen kann auch die vorliegende Erfindung Gebrauch machen.

II. FEHLERPRIORISIERUNG ZUR BEURTEILUNG DER GESUNDHEIT DER STEUERRECHNER (GESÜNDERER MASTER-CPU IST DIE PRIMARY)

[0039] Nach einer Weiterbildung der vorliegenden Erfindung ist nach einem weiteren Merkmal eine sogenannte Fehlerpriorisierung vorgesehen. Mit der sich daraus ergebenden technischen Lehre wird die Aufgabe gelöst, eine höhere Verfügbarkeit des redundanten Steuersystems ohne Datenverlust zu erreichen.

[0040] Bei der Fehlerpriorisierung handelt es sich darum, dass die möglichen Fehler, die in einer redundanten Steuerung auftreten können, gewichtet werden.

[0041] Auf den mindestens zwei Steuerrechnern erfolgt demnach eine Bewertung (Gewichtung) des Fehlerzustandes des jeweiligen Steuerrechners und in einer Vergleichseinheit werden die beiden von den Steuerrechnern ermittelten Fehler gegeneinander abgewogen, und in Abhängigkeit von dieser Abwägung wird vom einen auf den anderen Steuerrechner umgeschaltet.

[0042] Die Priorisierung nimmt entweder der Anwender vor oder es wird die (voreingestellte) Default-Priorisierung verwendet. So wird beim Auftreten eines Fehlers überprüft, ob die Partnersteuerung einen „gesünderen" Zustand hat und dann entschieden, ob ein Failover (Umschaltung zwischen dem einen Steuerrechner auf den anderen) durchgeführt wird. So hat der Anwender die Möglichkeit die redundante Steuerung für sein eigenes Einsatzgebiet zu konfigurieren.

[0043] Beispiel einer Fehlerpriorisierung:

[0044] Jeder Steuerrechner verwaltet eine zentrale Fehlerliste.

[0045] Sämtliche Fehler werden kategorisiert nach:

[0046] 1. Typ: Schweregrad des Fehlers

[0047] 2. Error = Einschränkung im Betrieb

[0048] 3. Warning = keine Einschränkung im Betrieb (Ausfallsicherheit nicht mehr gegeben)

[0049] FailLevel: Dieser Level bestimmt den Einfluss auf einen Failover

[0050] 0 = keinen Einfluss

[0051] 20 = Selbsttest nach n fehlerhaften Zyklen starten

[0052] 50 = Failover nach n fehlerhaften Zyklen durchführen

[0053] 100 = Failover sofort durchführen

[0054] StateLevel: Dieser Level bestimmt den Einfluss auf den lokalen Status

[0055] 0 = keinen Einfluss

[0056] 50 = geringen Einfluss → keine Änderung des Status (nur Warn-Meldung erzeugen)

[0057] 100 = großen Einfluss → Status ERROR

[0058] SyncLevel: Dieser Level bestimmt den Einfluss auf den Sync-Status

[0059] 0 = keinen Einfluss

[0060] 50 = geringen Einfluss → keine Änderung des Status (nur Warning)

[0061] 100 = großen Einfluss → Status ERROR oder INIT

Fehlernummer	Typ	FailLevel	StateLevel	SyncLevel	Beschreibung
BCR_E_FAIL	E	0	0	0	Unbekannter Fehler
BCR_E_RICONN	E	0	0	100	RI unterbrochen
BCR_E_RINETREDU	W	0	0	50	RI nicht redundant
BCR_E_LENGTH	E	0	0	100	BCP Länge zu klein
BCR_E_VERSION	E	0	0	100	BCR Protokollversion unterschiedlich
BCR_E_APPNAME	E	0	0	100	Applikationsname unterschiedlich
BCR_E_BLOCKNB	E	20	0	100	Unterschiedliche Anzahl Speicher-Blöcke
BCR_E_BLOCKDIFF	E	20	0	100	Unterschiedliche Speicher-Blöcke
BCR_E_UPDATE	E	0	100	0	SW-Update fehlgeschlagen
BCR_E_DEVCONN	E	100	100	0	Device Verbindung unterbrochen
BCR_E_DEVREDU	W	50	50	0	Device nicht redundant
BCR_E_DEVNETREDU	W	50	50	0	Device Verbindung nicht redundant
BCR_E_APPERROR1	W	50	50	0	Applikationsfehler1
BCR_E_APPERROR2	W	50	50	0	Applikationsfehler2
BCR_E_APPERROR3	W	50	50	0	Applikationsfehler3
BCR_E_APPFATAL	E	100	100	0	Fataler Applikationsfehler
BCR_E_APPDONE	E	50	100	0	Applikation schickt kein MIO_CMD_APPDONE
BCR_E_APPOVLOAD	E	0	100	0	Applikation Überlast
BCR_E_CYCLEOVLOAD	E	0	100	0	Zyklus Überlast
BCR_E_CYCLETIME	E	0	100	0	Zykluszeit nicht möglich
BCR_E_BCHTRAILER	W	0	0	0	BCH Trailer Fehler (z.B. Primary-Info wird nicht an BCH Slave übertragen)

[0062] Beispiel 1: Die RI-Verbindung ist unterbrochen (BCR_E_RICONN):

[0063] Keinen Einfluss auf den Failover (in dem Fall gar nicht möglich)

[0064] Keinen Einfluss auf den lokalen Status (LocaleState = RUN)

[0065] Synchronisierung nicht mehr gegeben (Sync = OFF)

[0066] Beispiel 2: Die RI-Verbindung ist nicht redundant (BCR_E_RINETREDU)

[0067] Keinen Einfluss auf den Failover

[0068] Keinen Einfluss auf den lokalen Status (LocaleState = RUN)

[0069] Die Synchronisierung funktioniert nach wie vor (Sync = ON); allerdings ist keine Ausfallsicherheit mehr geben → Warning

[0070] Beispiel 3: Die Device-Verbindung ist unterbrochen (BCR_E_DEVCONN):

[0071] Failover wird unmittelbar durchgeführt

[0072] (sofern auf der Secondary-CPU kein Fehler mit Level 100 vorliegt)

[0073] Der lokale Status wechselt nach ERROR

[0074] Die Synchronisierung funktioniert nach wie vor (Sync = ON)

[0075] Ablauf eines Failovers:

[0076] Der Failover wird immer von der Primary-CPU (das ist der gegenwärtig mit der Peripherie Daten austauschende Steuerrechner) ausgelöst. Er kann nur stattfinden wenn die RI Verbindung besteht.

[0077] In jedem Zyklus wird auf beiden CPUs der Fehler mit dem größten FailLevel und der zugehörigen Auftretenshäufigkeit ermittelt. Diese Information wird im zyklischen Frame an die jeweils andere CPU übertragen. Daraus ermittelt die Primary-CPU, ob sich die Secondary-CPU in einem besseren Zustand befindet und im Fehlerfall ein Failover ausgelöst werden soll.

[0078] Ein Failover wird ausgelöst, wenn...

[0079] • Auf der Primary-CPU ein Fehler mit FailLevel 100 vorliegt und auf der Secondary-CPU kein Fehler mit FailLevel 100 vorliegt.

[0080] • Auf der Primary-CPU ein Fehler mit FailLevel 50 und Häufigkeit > 75% vorliegt und auf der Secondary-CPU kein Fehler bzw. ein Fehler mit FailLevel < 50 vorliegt.

[0081] Zur Berechnung der Häufigkeit wird ein Algorithmus für einen gleitenden Mittelwert verwendet.

[0082] Ein weiteres Beispiel könnte so aussehen, dass der Anwender Punkte für die jeweiligen Fehler verteilt und der Steuerrechner mit den wenigsten Punkten ist die Primary-CPU und darf mit der Peripherie Daten austauschen.

[0083] Der Vorteil dieser Fehlerpriorisierung liegt darin, dass die Fehler benotet (priorisiert) werden können. Der Anwender kann dies für seine Belange konfigurieren und entscheiden, welchen Fehler er für wichtig erachtet. Solche Fehler bekommen dann höhere Priorität. Man schaltet also zwischen beiden Steuerrechnern immer hin und her, bis beide Steuerrechner tot sind bzw. den gleichen FailLevel haben.

[0084] Man bekommt damit eine sehr hohe Verfügbarkeit, obwohl der gegenwärtig mit der Peripherie Daten austauschende Steuerrechner fehlerhaft ist. Er ist aber gesünder als der andere Steuerrechner und kommuniziert so mit den Peripherie-Einheiten. Dies ist im Gegensatz zu bekannten Steuerrechnern nach dem Stand der Technik, wo einfach ein Fehler zum Umschalten auf den zweiten Steuerrechner führt, ein Zurückschalten ist aber nicht mehr möglich, weil der weg geschaltete Steuerrechner schon einen Fehler besitzt. Eine solche Fehlerpriorisierung zur Beurteilung der Fehlerfreiheit der Steuerrechner nach der Erfindung ist deshalb im Stand der Technik nicht bekannt.

III. MECHANISMUS ZUM SYNCHRONISIEREN DER MASTER-CPUS

[0085] Nach einem weiteren Merkmal der vorliegenden Erfindung ist ein Mechanismus zum Synchronisieren der Master-CPUs vorgesehen. Mit der sich daraus ergebenden technischen Lehre wird ebenfalls - wie im vorherigen Abschnitt - die Aufgabe gelöst, ohne Datenverlust an der Peripherie eine Umschaltung zwischen redundant laufenden Steuerrechnern zu erreichen, komplette andere Speicher, so ist gewährleistet, dass beim Fehler eines Speichers nicht alle Applikationen betroffen sind.

[0086] Des Weiteren wird mit diesem Mechanismus gewährleistet, dass die redundanten Steuerungen immer redundant werden und nicht den „Grünen Tod" sterben. Grüner Tod bedeutet, dass beide Steuerungen keinen Fehler haben, aber sich nicht synchronisieren, so nicht von einer redundanten Steuerung geredet werden kann. Der Grüne Tod tritt oft während der Bootphase auf. Des Weiteren wird mit dem Mechanismus verhindert, dass die eingestellte Zykluszeit der Applikation verletzt wird.

[0087] Es handelt sich hierbei um die Steuerungen, die in jeweils einem der Steuerrechner 1 und 2 angeordnet sind.

[0088] Erfindungsgemäß wird ein Hot-Standby-System verwendet, bei dem die Master-CPUs in den zueinander redundanten Steuerrechnern synchronisiert werden.

[0089] Unter „synchronisiert" wird verstanden, dass die benötigte Software (Betriebssystem, Treiber, Dienste, Applikation usw.) der Master-CPUs und die Prozessvariablen der redundanten Applikationen zwischen den Master-CPUs der zueinander redundanten Steuerrechner abgeglichen werden. Der Abgleich der Prozessvariablen findet zyklisch statt und der Abgleich der Software wird beim Boot-Vorgang durchgeführt.

[0090] Bei einem Synchronisationsvorgang ist die Problematik, dass sich ständig auch die Werte ändern, die gerade abgeglichen wurden und die Steuerung so in ein Deadlock kommt. Eine weitere Problematik ist, dass durch die Synchronisation die vorgegebene Zykluszeit der Applikation überschritten wird. Diese Erfindung beschreibt ein Verfahren, wie diese beiden Problematiken gelöst werden.

SYNCHRONISATION DER MASTER-CPUS IN DEN BEIDEN ZUEINANDER REDUNDANTEN STEUERRECHNERN:

[0091] Die jeweilige Applikation auf der Primary-CPU und die hierzu redundante Applikation auf der Secondary-CPU melden ihren Speicher an, der redundant gehalten werden muss. Dieser Speicher wird in Blöcke (z.B. max. Größe 1400Bytes) unterteilt. Der Datenabgleich der redundanten Steuerrechner erfolgt durch die Übertragung der einzelnen Datenblöcke zum anderen Steuerrechner. Um kleine Zykluszeiten zu realisieren wurde ein Mechanismus entwickelt, die Blöcke in mehreren Zyklen zu übertragen.

[0092] Zuerst sendet die Primary-CPU über die Redundancy Interconnect immer die Blöcke, die sich über mehrere Zyklen nicht geändert haben und in den folgenden Zyklen die Blöcke, welche sich öfters ändern. In einer ersten Ausführung speichert die Secondary-CPU alle empfangenen Blöcke in einem Cache. Wurden alle Blöcke des Speicherbereichs empfangen, wird der Speicher übernommen. Enthält der Cache veraltete Blöcke (Daten aus vorigem Zyklus) so wird der entsprechende Block verworfen.

[0093] In einer zweiten Ausführung entfällt der Cache und die von der Primary-CPU gesendeten Daten werden direkt in den Arbeitsspeicher der Secondary-CPU geschrieben.

[0094] Der Vorteil beider Maßnahmen ist, dass durch den Mechanismus zur Synchronisierung die Zykluszeiten, also die Zykluszeiten der Applikationen, nicht verletzt werden, weil eine Aufteilung in kleinen (z.B. nur einige wenige KBit-lange) Blöcke erfolgt und diese Blöcke schnell übertragen werden können.

[0095] Des Weiteren ist der Mechanismus so ausgebildet, dass die Blöcke, die sich öfter ändern, zum Schluss übertragen werden, so dass gesichert ist, dass das Applikationsprogramm nicht den sogenannten „grünen Tod" stirbt. Das bedeutet, dass der vollständige Redundantenmodus niemals erreicht wird, obwohl die Steuerungen keinen Fehler aufweisen. Die Erfindung ist so ausgereift, dass der Redundantenmodus immer erreicht wird.

[0096] Das Kernelement einer redundanten Steuerung ist demnach der Mechanismus zum Synchronisieren der Master-CPUs in den beiden Steuerrechnern 1 und 2.

[0097] Man unterscheidet zwischen zwei Synchronisierungen:

[0098] Während der Bootphase wird die Software synchronisiert. Das bedeutet, die Synchronisation des Betriebssystems, der Treiber, der Applikationen usw.. Des Weiteren ist im zyklischen Betrieb die Synchronisierung so zu verstehen, dass die Prozessvariablen der einzelnen Applikationen zwischen den CPUs der beiden redundanten Steuerrechner synchronisiert werden.

[0099] Der Mechanismus der Synchronisation ist der gleiche:

[00100] Die Daten, die zur Synchronisation bereitstehen, werden in kleine Blöcke aufgeteilt.

Diese Blöcke werden innerhalb der vorhandenen Zykluszeit, die der Anwender angibt, übertragen.

[00101] Das Problem besteht darin, dass wenn sich mehrere Blöcke oder Variablen im zyklischen Task öfters ändern, würde man immer diesen Block wiederholt, übertragen und so nie redundant werden. Anhand des erfindungsgemäßen Synchronisationsmechanismus wird bewertet, wie viele Blöcke sich ändern und welche von den sich ändernden Blöcken die größte Änderungsfrequenz haben. Diese Blöcke werden dann bei der in einem Zyklus statt findender Übertragung zurück gestellt und als letztes übertragen. Man erreicht so idealerweise einen redundanten Modus. Damit wird vermieden, dass man niemals redundant werden kann, weil sich zu viele Variable ändern und der Zyklus der Applikation nicht verletzt wird

IV: DATENAUSTAUSCH ZUM IO-DEVICE MIT UMSCHALTUNG IM GLEICHEN ZYKLUS. DIESES FÜHRT ZU EINEM STOßFREIEN UMSCHALTEN

[00102] Nach einer Weiterbildung der vorliegenden Erfindung ist vorgesehen, dass der Datenaustausch zu der Peripherie mit einer eine „Null-Millisekunde" dauernden Umschaltung geschieht, d. h. ohne Zeitverzug und hierdurch eine betriebsbedingte hohe Geschwindigkeit erreicht wird.

[00103] Bei einer Hot-Standby-Redundanz ist bei Auftreten eines Fehlers die stoßfreie Umschaltung von der einen Master-CPU (z.B. der Primary) auf die andere Master-CPU (z.B. der Secondary) die zentrale, angestrebte Funktion.

[00104] Um ein wirklich stoßfreies Umschalten zu garantieren, schreiben die beiden Master-CPUs aktiv die Daten zu den IO-Devices (Peripherie-Einheiten). Der Anwender kann konfigurieren, welche Daten das IO-Device zu den Applikationen weiterleiten soll.

[00105] Folgende Möglichkeiten bestehen:

[00106] 1. Es werden grundsätzlich die Werte von der Primary-CPU weitergeleitet, wenn das IO-Device keine Daten von der Primary-CPU bekommt, werden automatisch die Werte von der Secondary-CPU weitergeleitet. Nach wie viel Zyklen diese automatische Umschaltung erfolgt soll, ist konfigurierbar.

[00107] 2. Der Anwender kann das IO-Device so konfigurieren, dass aus den beiden Daten von den beiden Master-CPU ein Mittelwert gebildet wird, der zur Applikation weitergeleitet wird.

[00108] 3. Der Anwender kann zusätzlich zu den Daten der Master-CPU einen weiteren Wert konfigurieren und dann wird ein Mehrheitsentscheid durchgeführt.

[00109] 4. Der Anwender hat die Möglichkeit, seinen eigenen Voter (Entscheidungsfinder) zu schreiben und kann so selbst die Regeln definieren, welche Werte zur Applikation weitergeleitet werden sollen.

[00110] 5. Der Anwender hat die Möglichkeit einen Wertebereich zu definieren

[00111] 6. Der Anwender hat die Möglichkeit den mittleren Wert auszuwählen

[00112] Man erreicht für die erfindungsgemäße redundante Steuerung eine höhere Verfügbarkeit und des Weiteren eine „Null-Millisekunden" dauernde Umschaltzeit. Das bedeutet, dass wenn der zweite Rechner abgeschaltet wird, der andere Rechner in Null-Millisekunden aktiv wird. Aus der Sicht der Peripherie bedeutet dies, dass die Peripherie einen solchen Umschaltstoß zwischen den Steuerrechnern nicht feststellen kann. Es wird demnach eine stoßfreie Umschaltung generiert, so dass die Peripherie ständig von einer Hauptsteuerung Daten bekommt und so permanent lauffähig ist.

[00113] Der Anwender hat die Möglichkeit seinen eigenen Voter (Entscheidungsfinder) zu schreiben und sich selbst die Regeln zu definieren, welche Werte von der im Steuerrechner ablaufenden Applikation zur Peripherie weitergeleitet werden.

[00114] Ein mit digitalen Werten arbeitenden Voter in der Peripherieeinheit trifft zum Beispiel eine „zwei-aus-drei" Entscheidung. Bei Vorhandensein von drei digitalen Werten an drei Eingängen der Peripherie-Einheit, wird entschieden, welche zwei digitalen Werte (z.B. „1" oder „0") identisch an den drei digitalen Eingängen aufgetreten sind. Dieser eine digitale Wert (z.B. eine logische „1") wird dann zum Maschinenteil der Peripherie-Einheit weitergeleitet und in ein logisches Signal zur Ansteuerung eines Gerätes (z.B. einer Pumpe) genutzt.

[00115] Des Weiteren kann nach der gleichen Erläuterung ein (analoger) Voter einen Mittelwert aus zwei oder mehreren Analogwerten bilden, die an den Eingängen der Peripherie-Einheit anliegen. Der aus der Mittelwertbildung oder einer anderen, die analogen Werte an den Eingängen zusammen fassenden Rechenoperation stammende Ergebnis-Analogwert wird dann an das Maschinenteil der Peripherie-Einheit weiter geleitet.

V. PARALLELBETRIEB EINER REDUNDANTEN UND EINER NICHT-REDUNDANTEN APPLIKATION PARALLEL AUF EINER STEUERUNG LAUFFÄHIG.

[00116] Als weiteres Merkmal der vorliegenden Erfindung ist vorgesehen, dass eine oder mehrere redundante Applikationen mit einer oder mehreren nicht-redundanten Applikationen parallel auf jedem der Steuerrechner lauffähig sind. Bei einer Multitask-Applikation kann ein oder mehrere Tasks redundant sein und im gleichen Projekt ein oder mehrere Tasks nicht-redundant sein.

[00117] Hierunter wird verstanden, dass im Falle einer redundanten Steuerung die auf den beiden Steuerrechnern ablaufenden Applikationen zueinander identisch sind und unter gleichen Zeit- und Datenbedingungen auf beiden Steuerrechnern identisch ablaufen.

[00118] Es kann jedoch vorgesehen sein, dass auf jedem Steuerrechner ein oder mehrere Applikationen ablaufen, die nicht auf dem anderen Steuerrechner abgebildet sind. Sie sind also nicht zueinander redundant.

[00119] Solche isoliert auf dem einen oder anderen Steuerrechner laufenden nicht-redundanten Applikationen arbeiten mit einer zugeordneten Peripherie-Einheit zusammen. D. h. eine nicht redundante Applikation läuft dann selbstständig auf einem einzigen Steuerrechner, während auf dem anderen Steuerrechner eine derartige Applikation nicht vorhanden ist, jedoch möglicherweise eine andere nicht-redundante Applikation.

[00120] Merkmal hierbei ist, dass redundante Applikationen (Applikationen, die auf beiden Steuerungen laufen und synchronisiert werden) und nicht redundante Applikationen gleichzeitig auf einer Steuerung laufen können.

[00121] Hieraus folgt, dass der Anwender nicht alle Applikationen als redundante Applikationen definieren muss und dadurch einen geringeren Synchronisationsaufwand hat, d. h. dass der Datenverkehr zwischen der primären und sekundären CPU verringert wird und so geringere Zykluszeiten, d. h. schnellere Zykluszeiten generiert werden können.

[00122] Vorteil der Maßnahme ist ferner, eine nicht-redundante Applikation auf einem Steuerrechner ablaufen lassen zu können, weil man ein bestimmtes Rechenpotential zur Verfügung hat, welches nicht unbedingt redundant vorgehalten werden muss, so dass man die Steuerrechner auch noch für andere, nicht-redundante Überwachungsaufgaben einsetzen kann und die Rechenkapazität eines oder beider Steuerrechner für dort ablaufende nicht-redundante Applikationen nutzen kann.

[00123] Damit wird der Einsatzbereich derartiger redundanter Steuerrechner wesentlich erweitert.

[00124] Im Folgenden wird die Erfindung anhand von mehrere Ausführungswege darstellenden Zeichnungen näher erläutert. Hierbei gehen aus den Zeichnungen und ihrer Beschreibung weitere erfindungswesentliche Merkmale und Vorteile der Erfindung hervor.

[00125] Figur 1: Schematisierte Darstellung eines redundanten Steuerungssystems nach der Erfindung

[00126] Figur 2: Ein Ausschnitt aus dem Steuerungssystem der Figur 1 mit Darstellung der beiden redundanten Steuerrechner

[00127] Figur 3: Die gleiche Darstellung wie Figur 2 bei Ausfall einer Applikation

[00128] Figur 4: Schematisiert im Blockschaltbild die Fehler-Priorisierung zwischen den beiden Steuerrechnern in einem ersten Schaltzustand

[00129] Figur 5: Die gleiche Darstellung wie Figur 4 in einem zweiten Schaltzustand

[00130] Figur 6: Die gleiche Darstellung wie Figur 4 in einem dritten Schaltzustand

[00131] Figur 7: Die Synchronisation der Steuerrechner in der Art eines Blockschaltbildes

[00132] Figur 8: Der Zeitablauf der Übertragung der verschiedenen Variablen von einem Steuerrechner zum anderen

[00133] Figur 9: Eine weitere Darstellung des Zeitablaufes bei der Übertragung der Variablen

[00134] Figur 10: Das Blockschaltbild eines digitalen Voters

[00135] Figur 11: Blockschaltbild eines analogen Voters

[00136] Figur 12: Die prinzipielle Darstellung der Verschaltung der beiden redundanten Steuerrechner in der Art nach dem Blockschaltbild der Figuren 1 und 2

[00137] Figur 13: Ein Blockschaltbild der Entscheidungsstufe, in der entschieden wird, auf welchen der Steuerrechner umgeschaltet wird

[00138] Figur 14: Ein gegenüber Figur 1 und 2 erweitertes Blockschaltbild, welches darstellt, das neben den zueinander redundanten Applikationen auch nicht redundante Applikationen auf beiden Steuerrechnern ablaufen können

[00139] In den Figuren 1 und 2 ist allgemein dargestellt, wie die Zusammenschaltung der Steuerrechner nach den Figuren 2 und 3 in einem größeren Zusammenhang verschaltet sein könnte.

[00140] Hierbei sind die beiden zueinander redundanten Steuerrechner 1, 2 in einem Fabrik-Netzwerk 10 eingebunden, an dem eine Reihe von weiteren Stationen angeschlossen sein können, wie zum Beispiel eine Station für HMI/Scad oder eine Station für die benutzerseitige Programmierung der gesamten Anlage nach Figur 1. Es liegt auf der Hand, dass das Fabrik-Netzwerk 10 auch vollkommen entfallen kann oder durch andere Steuerungsmittel ersetzt werden kann, weil es nur darum geht, die Konfiguration der beiden zueinander redundanten Steuerrechner 1, 2 einzustellen und ggf. zu steuern.

[00141] Wesentlich ist jedenfalls, dass zwischen den beiden Steuerrechnern eine Busleitung 3, 4 angeordnet ist, die auch später als RI-Verbindung bezeichnet wird, was bedeutet, dass auch diese Verbindungsleitung zwischen den beiden Steuerrechnern 1, 2 redundant ausgebildet ist, sodass wenn die eine Busleitung 3 versagt, der gesamte redundante Datenverkehr zwischen den Steuerrechnern 1, 2 über die noch verbleibende Busleitung 4 abgewickelt wird.

[00142] In der bevorzugten Ausführungsform ist vorgesehen, dass auf beiden Steuerrechnern 1, 2 die gleichen Applikationen laufen, die genau zeitgleich und identisch ablaufen, sodass der eine Steuerrechner 1 das gespiegelte Abbild des anderen Steuerrechners ist. Beide Steuerrechner 1, 2 arbeiten über die Busleitungen 5, 6 auf Peripherie-Einheiten 7, 8, 9, die nur beispielhaft dargestellt sind. Es können mehr oder weniger Peripherie-Einheiten vorhanden sein und es reicht für die Erläuterung der vorliegenden Verbindung auch aus, wenn nur eine einzige Peripherie-Einheit vorhanden ist.

[00143] Der Verbindungsbus mit den Busleitungen 5, 6 wird auch später als IO-Network bezeichnet und kann zum Beispiel ein RT Ethernet A- oder ein RT Ethernet B-Netzwerk sein.

[00144] Die Peripherie-Einheiten 7-9 sind sogenannte IO-Devices und arbeiten als Slaves, was bedeutet, dass sie Werte entgegen nehmen, und an eine Maschinensteuerung weiter geben.

[00145] Zum Beispiel erhält Peripherie-Einheit 7 über die Busleitung 5, 6 den Befehl, einen bestimmten Ausgang auf einen bestimmten Wert zu setzen und gleichzeitig einen Eingang auszulesen. Ebenso sind solche Befehle möglich, dass eine bestimmte Lampe, ein Pumpenparameter oder andere Maschinenparameter durch die jeweilige Peripherie-Einheit 7-9 gesetzt werden.

[00146] Es ist nicht dargestellt, das die Peripherie-Einheiten 7-9 ihrerseits auf einen I/O-Bus arbeiten können, über den die Maschinensteuerungen angeschlossen sind, denen die Werte von Seiten der Peripherie-Einheiten 7-9 zugeleitet werden.

[00147] Wichtig bei der vorliegenden Erfindung ist gemäß Figur 2 nun, dass auf jedem der Steuerrechner 1, 2 zueinander identische Applikationen zum gleichen Zeitpunkt und mit gleichen Werten laufen, sodass die Applikationen 11, 12, 13 auf dem einen Steuerrechner 1 genau identisch mit den gleichen Werten und den gleichen Variablen als Applikationen 11', 12', 13' auf dem zweiten Steuerrechner 2 ablaufen.

[00148] Alle Applikationen liefern Werte und arbeiten auf die gemeinsame Busleitung 5, 6, sodass alle Werte zunächst über die Busleitung 5, 6 an den Eingängen der Peripherie-Einheiten 7, 8, 9 anliegen.

[00149] In jeder Peripherie-Einheit 7, 8, 9 liegen nun die Ergebnisse der Rechenoperationen der einzelnen Applikationen 11, 11', 12, 12', 13, 13' in Form von Datenresultaten 14, 15, 16 vor.

[00150] Ein solches Datenresultat 14, 15, 16 kann beispielsweise darin bestehen, dass die Peripherie-Einheit auf Grund des Datenresultates 14 von der Applikation 11 den Befehl bekommt, einen bestimmten Eingang zu lesen oder einen Ausgang auf einen bestimmten Wert zu setzen.

[00151] In gleicher Weise arbeitet die Peripherie-Einheit 8 mit der Applikation 12 im Steuerrechner 1 zusammen und gibt dort die Datenresultate 15 als Ergebnis der Rechenoperation der Applikation 12 auf einem daran angeschlossenen I/O-Ausgang aus.

[00152] In gleicher Weise empfängt die Peripherie-Einheit 9 die Daten von der Applikation 13 als Datenresultate 16 und steuert dementsprechend die angeschlossenen Maschinenelemente an.

[00153] Das gezeigte Blockschaltbild nach Figur 2 zeigt dem gemäß einen Funktionszustand der zueinander redundanten Steuerrechner 1, 2 in der Weise, dass alle Applikationen in Steuerrechner 1 „gesund" sind und deshalb nur befugt sind, mit den Peripherie-Einheiten 7, 8, 9 Werte auszutauschen oder Werte an diese Peripherie-Einheiten 7-9 zu liefern, wobei die Werte von den gleichzeitig mitlaufenden Applikationen 11', 12', 13' im Steuerrechner 2 verworfen werden und keinen Einfluss auf die Peripherie-Einheiten 7, 8, 9 haben.

[00154] Das Blockschaltbild 2 geht also davon aus, dass der primäre Steuerrechner 1 völlig gesund ist und er ist deshalb nicht auf ein Eingreifen des sekundären Steuerrechners 2 angewiesen.

[00155] Es kann im Übrigen durch Programmeingriff von vornherein bestimmt werden, welcher der Steuerrechner 1 immer zunächst als primärer Steuerrechner angesehen wird und welcher der Steuerrechner 1 oder 2 zu Beginn aller Rechenoperationen als sekundärer Steuerrechner angesehen wird.

[00156] Figur 3 zeigt nun einen Fehlerfall im primären Steuerrechner 1, wo erkennbar ist, dass zwar die Applikationen 11 und 12 weiter laufen, dass aber die Applikation 13 zum Stillstand gekommen ist.

[00157] In diesem Fall wird ein Umschaltimpuls 42 über die redundante Busleitung 3, 4 zum anderen Steuerrechner 2 geschickt und dieser wird nun ermächtig, die parallel zu der stillgelegten Applikation 13 gespiegelten Applikation 13' weiter zu betreiben, wobei ohne Umschaltung zwischen dem Steuerrechner 1 und 2 und ohne Datenverlust die Applikation 13' die Rechenoperation weiterführt, ohne dass in der Peripherie-Einheit 7, 8, 9 bemerkt wird, dass die Applika-

tion 13 im Steuerrechner 1 zum Erliegen gekommen ist.

[00158] Erreicht wird dies durch die den Peripherie-Einheiten vorgeschalteten Entscheidungsstufen 17, 18, 19, wobei bei der Peripherie-Einheit 7 die Entscheidungsstufe 17 maßgebend ist und diese Entscheidungsstufe stets sowohl die Daten von der Applikation 11 als auch die Daten von der Applikation 11' synchron erhält und auch die Daten an beide Steuerrechner 1 und 2 verschickt.

[00159] In dieser Entscheidungsstufe wird entschieden, welche Daten nun tatsächlich als Datenresultat 14 in die Peripherie-Einheit 7 übergeben werden, wie dies durch den vertikalen Pfeil in Figur 3 in jeder Peripherie-Einheit 7, 8, 9 symbolisiert ist.

[00160] Gleiches gilt auch für die Peripherie-Einheit 8, wo in der Entscheidungsstufe 18 stets synchron die Werte von der Applikation 12 und 12' ankommen, was mit den Bezugszeichen XApp2 und XApp2' bezeichnet ist.

[00161] In der Entscheidungsstufe 18 wird wiederum entschieden, ob die Werte XApp2 oder XApp2' übernommen werden, wobei im gezeigten Ausführungsbeispiel die Werte XApp2 übernommen werden, d. h. die Werte, die von der ungestört laufenden Applikation 12 in Steuerrechner 1 erzeugt wurden.

[00162] Auf Grund des Fehlers der Applikation 13 im Steuerrechner 1 wird nun in der Entscheidungsstufe 19 in der Peripherie-Einheit 9 entschieden, dass nunmehr sofort und ohne Datenverlust zu jedem beliebigen Zeitpunkt auf die Applikation 13' (App3') umgeschaltet wird und die von dieser App3' erzeugten Werte werden zu den Werten XApp3' in der Peripherie-Einheit 9 verarbeitet. Grund für das Umschalten auf die Applikation 13' (App3') im redundanten Steuerrechner 2 ist, dass die Applikation 13 (App3) im Steuerrechner 1 zum Erliegen gekommen ist.

[00163] Auf diese Weise arbeitet die Peripherie-Einheit 9 mit den Werten XApp3' ungestört und ohne Datenunterbrechung aufgrund der fehlerfreien Applikation 13' im Steuerrechner 2 weiter, ohne dass es zu einem Datenverlust in der Peripherie-Einheit kommt.

[00164] Aus dem Vergleich der Blockschaltbilder 1, 2 und 3 ergibt sich der wesentliche Vorteil der Erfindung, nämlich, dass wenn ein oder mehrere Applikationen in einem Steuerrechner 1, 2 zum Erliegen kommen, dass dann die anderen Applikationen im gleichen Steuerrechner noch weiter laufen, ohne dass es dort bereits schon zu einer Umschaltung zwischen dem ersten und dem zweiten Steuerrechner kommen müsste. Im Stand der Technik ist nicht vorgesehen, dass mehrere Applikationen von einem Steuerrechner verarbeitet werden.

[00165] Ein weiteres Beispiel: Die Applikationen 11 und 12 (App1 und App2) im Steuerrechner 1 sind ausgefallen, nur die Applikation 13 (App3) im Steuerrechner 1 läuft noch ungestört. Es läuft dann der Steuerrechner 1 mit seiner Applikation 13 weiter und liefert die Daten XApp3 an die Peripherie-Einheit 3.

[00166] In einer anderen Ausführung kann vorgesehen sein, dass dieses Verhalten vom Anwender konfigurierbar ist. Der Anwender hat die Möglichkeit zu konfigurieren, ob der Steuerrechner 2 auch die nicht defekte Applikation (in diesem Fall die Applikation 13) übernehmen soll.

[00167] Bezüglich der auf dem Steuerrechner 1 ausgefallenen Applikationen 11 und 12 (App1 und App 2) wird auf den Steuerrechner 2 umgeschaltet und die auf diesem Steuerrechner noch ungestört laufenden, identischen Applikationen 11' und 12' (App 1' und App2') liefern ihre Werte XApp1' und XApp2' an die Peripherie-Einheiten 7 und 8, ohne dass diese Peripherie-Einheiten 7 und 8 merken, dass von den fehlerbehafteten Applikationen 11 und 12 auf die fehlerfreien Applikationen 11' und 12' umgeschaltet wurde.

[00168] Es liegt demgemäß eine wesentlich höhere Verfügbarkeit der Steuerrechner 1, 2 vor und es ist kein Datenverlust bei der Umschaltung von einer zum Erliegen gekommenen ersten Applikation auf eine zweite Applikation zu befürchten, weil beide Applikationen identische und synchrone Werte zu den Peripherie-Einheiten liefern und erst dort in einer Entscheidungsstufe 17-19 festgestellt und entschieden wird, welche Werte nun von welcher Applikation tatsächlich

in die Peripherie-Einheit übernommen werden.

II. FEHLERPRIORISIERUNG

[00169] In Erfüllung der gestellten Aufgabe, eine Umschaltung zwischen Steuerrechnern 1, 2 ohne Datenverlust vorzunehmen, dient auch die in den Figuren 4-6 dargestellte Fehlerpriorisierung, was bedeutet, dass eine Umschaltung von einem Steuerrechner auf den anderen nur dann erfolgt, wenn der eine Steuerrechner krank ist und der andere Steuerrechner gesünder als der erste Steuerrechner ist.

[00170] Es werden sozusagen Noten oder Fehler über den Gesundheitszustand der einzelnen Steuerrechner vergeben und durch die Notenvergabe wird sichergestellt, dass Steuerrechner auch weiterlaufen, wenn deren „Gesundheitszustand angeschlagen" ist, wobei ein Weiterlaufen dieses Steuerrechners nur dann vorgesehen ist, wenn der andere Steuerrechner noch kränker als der erste Steuerrechner ist.

[00171] Durch diese Fehlerpriorisierung wird eine durchgehende Verfügbarkeit der einzelnen Steuerrechner erreicht, was sich im Hinblick auf die vorher genannten Ausführungsbeispiele nach den Figuren 1-3 nahtlos ergänzt.

[00172] Im gezeigten Ausführungsbeispiel nach den Figuren 4-6 wird zunächst vorausgesetzt, dass ein Fehler mit der Fehlerstufe Prio 1 ein hochwertiger (schwerer) Fehler ist, wenn ein Fehler beispielsweise mit der Stufe Prio 3 ein niederwertiger (also leichterer) Fehler ist.

[00173] Ausgehend von dem Blockschaltbild nach Figur 3 werden die beiden Steuerrechner 1, 2 nur noch schematisiert dargestellt und es ist erkennbar, dass die vorher als die Applikationen 11, 12, 13 dargestellten unabhängig voneinander ablaufenden Maschinenprogramme nun einfach nur noch als App 1, App 2, App 3 bezeichnet wurden.

[00174] Alle diese Applikationen laufen auf dem Steuerrechner 1 ab und es wird jetzt angenommen, dass ein niederwertiger Fehler der Stufe 3 in Steuerrechner 1 passiert, der auch als Master 1 bezeichnet wird.

[00175] Demgemäß wird ein Umschaltvorgang 23 in Master 1 ausgelöst und eine Umschaltung über die redundante Busleitung 3, 4 auf den zweiten Master 2 (Steuerrechner 2) ausgeführt, sodass dieser Umschaltvorgang 24 dafür sorgt, dass der zweite Steuerrechner 2, der vorher der Secondary-Master war und der Primary-Master ist und nun befugt ist, zu leiten.

[00176] Die Figur 5 zeigt den umgekehrten Fehlerfall, wo erkennbar ist, dass in Master 2 ein höherwertiger Fehler 2 aufgetreten ist. Deshalb wird im Umschaltvorgang 24 dieser vorher als Primary-Master arbeitende Steuerrechner 2 nun als Secondary-Master zurückgestuft und über die redundante Busleitung 4, 5 wird der vorher als Secondary-Master arbeitende Steuerrechner 1 nun zum Primary-Master ernannt, obwohl er einen niederwertigeren Fehler 3 aufweist.

[00177] Hieraus ergibt sich, dass der Steuerrechner 2 kränker ist, als vergleichsweise der Steuerrechner 1, sodass hiermit automatisch von den vorher als Primary angegebene Steuerrechner 2 auf den einen geringeren Fehler aufweisenden Steuerrechner 1 umgeschaltet wird und dieser nun befugt ist, über die Busleitung 5, 6 Daten an die Peripherie zu liefern.

[00178] In Figur 6 ist als weiteres Ausführungsbeispiel dargestellt, dass wenn beide Steuerrechner 1, 2 eine gleiche Fehlerstufe, zum Beispiel die Fehlerstufe III, aufweisen, dass dann keine Umschaltung erfolgt. Es ist bei den Umschaltvorgängen 23, 24 dargestellt, dass nicht umgeschaltet wird und beide Steuerrechner 1, 2 - obschon fehlerbehaftet - weiter arbeiten, wobei der vorher als Primary-Master definierte Steuerrechner 1 auch weiterhin als Primary-Master arbeitet und über seinen Ausgang auf die Busleitung 5, 6 Daten liefert.

[00179] In den beschriebenen Beispielen ist es selbstverständlich, dass der Anwender über sämtliche Fehlerereignisse und Status informiert wird.

[00180] Zur Erfindung der gleichen Aufgabe, nämlich der stoßfreien Umschaltung zwischen den Steuerrechnern ohne Datenverlust und gleichzeitige Erhöhung der Verfügbarkeit dient auch

die nachfolgende Beschreibung:

III. SYNCHRONISATION DER STEUERRECHNER

[00181] Gemäß den Figuren 7-9 wird dargestellt, wie sichergestellt werden kann, dass die beiden Steuerrechner 1, 2 stets über die redundante Busleitung 3, 4 auf dem gleichen Informationsstand gehalten werden können ohne dass sie am „Grünen Tod" sterben und die Zykluszeiten minimiert werden können. Damit wird vermieden, dass der gesamte Datenverkehr zum Erliegen kommt.

[00182] Hierbei sind in Figur 7 verschiedene Datenblöcke A, B, C und D dargestellt. Ferner ist in jedem Kästchen für die Datenblöcke A, B, C und D durch einen Pfeil dargestellt, welche Änderungsgeschwindigkeit die dort in den Datenblöcken vorgehaltenen Variablen haben.

[00183] Durch den langen Pfeil im Datenblock A wird dargestellt, dass die Variablen V1 und V2 sich schnell ändern und die Variable V3 keine Änderung erfährt, sodass insgesamt eine starke Änderung dieses Datenblockes A gegeben ist.

[00184] Der Datenblock B ist eine geringere Änderung der Variablen V4, V5 und V6 gegeben, weil sich nur die Variable V5 während eines Rechenzyklus ändert.

[00185] Im Datenblock C erfolgt keine Änderung der dort angegebenen Variablen V7, V8 und V9, während im Datenblock D nur nicht zueinander redundante Variablen X1, X2, X3, vorgehalten werden, die keinen Abgleich benötigen. Deshalb werden diese Variablen, die nicht redundant vorgehalten werden müssen, auch nicht zum anderen Steuerrechner übertragen.

[00186] In Schraffur ist im Übrigen angegeben, dass nur die Datenblöcke A, B und C redundant gehalten werden müssen, während dies in Datenblock D nicht der Fall ist.

[00187] Erfindungsgemäß ist nun vorgesehen, dass bei einer seriellen Übertragung über die Busleitung zunächst sich die am wenigsten ändernden Datenblöcke, nämlich die Variablen V7, V8 und V9 des Datenblocks C in der ersten Übertragungszeit t1 übertragen werden, danach folgend die Variablen des Datenblocks B, weil sich diese Variablen schneller ändern, als die Variablen des Datenblocks C und danach erst die Variablen des Datenblocks A übertragen werden, weil sich diese am schnellsten ändern.

[00188] Bei der Übertragung dieser drei Datenblöcke zur Übertragungszeit t1, t2 und t3 sind zwei verschiedene Ausführungsformen vorgesehen.

[00189] Der rechte Teil in Figur 7 zeigt, dass diese Datenblöcke erst in einem Cache-Bereich aufgenommen werden können und dann „in einem Rutsch" in den Speicher des Steuerrechners 2 geschrieben werden können.

[00190] In einer anderen Ausführungsform kann es jedoch vorgesehen sein, dass der Cache-Bereich entfällt und die übertragenen Datenblöcke A, B, C direkt in den Speicher des Steuerrechners 2 geschrieben werden.

[00191] Wie eingangs ausgeführt, müssen die Datenblöcke D nicht übertragen werden. Die Figur 8 zeigt die nacheinander folgende Übertragung zu den Zeiten t1, t2 und t3 und die Figur 9 zeigt, dass bezüglich einer Zykluszeit für jede Applikation, wie zum Beispiel für eine Dauer von 10 ms angesetzt wird. Der Applikationszyklus wird durch diesen Mechanismus nicht verletzt, da die noch verbleibende Zeit für die Synchronisation verwendet wird. In dem Beispiel benötigt die Applikation 8ms, wobei ein Zyklus von 10ms eingestellt wurde. So bleiben für die Synchronisation noch 2ms übrig.

IV. VOTER

[00192] Eingangs wurde anhand der Blockschaltbilder der Figuren 1-3 erläutert, wer entscheidet, welche Peripherie-Einheiten 7, 8, 9 mit welchen Werten arbeiten darf. Aus diesem Grund wird in Figur 3 angegeben, dass den Peripherie-Einheiten 7, 8, 9 Entscheidungsstufen 17, 18, 19 vorgeschaltet sind, in denen entschieden wird, welche Daten von welcher Applikation in die

Peripherie-Einheit 7, 8, 9 übernommen werden. Diese Entscheidungsstufe wird nachfolgend auch Voter genannt, wobei es gemäß Figur 10 einen Voter (Entscheidungsstufe) für die Entscheidung der Übernahme digitaler Werte gibt, während in Figur 11 eine solche Entscheidungsstufe 17-19 in Form eines analogen Voters vorgestellt wird, der unter Aufnahme von analogen Werten entscheidet, welcher analoge Wert in die Peripherie-Einheit 7-9 eingespeist wird.

[00193] Im Ausführungsbeispiel nach Figur 10 bezüglich des digitalen Voters wird angegeben, dass zum Beispiel der Parameter P I den digitalen Wert 1 aufweist und am Eingang 27 die Entscheidungsstufe 17-19 anliegt, während am Eingang 28 ein Schalter S 1 den digitalen Wert 0 aufweisen kann.

[00194] Ferner besteht die Möglichkeit, dass über einen Eingang 29 eine dritte Variable C I mit dem digitalen Wert 1 eingespeist wird.

[00195] In den Entscheidungsstufen 17 bis 19 wird nun eine sogenannte II OUT OF III-Entscheidung durchgeführt, d. h., wenn zweimal der digitale Wert 1 und einmal der digitale Wert 0 vorkommt, wird der digitale Wert 1 gemäß der Übergabe 30 in die Peripherie-Einheit 7-9 als Datenresultat 14-16 übernommen. Somit wird zum Beispiel ein Kanal I auf den digitalen Wert 1 gesetzt. Dieser Wert Ch I = 1 wird an dem daran anschließenden Maschinenteil 26 übergeben.

[00196] Die gleichen Verhältnisse gelten auch für eine analoge Entscheidungsstufe nach Figur 11.

[00197] Dort ist angegeben, dass über den Eingang 27 der analoge Wert 1,23 und über den Eingang 28 der analoge Wert 1,44 anliegt. Bei einem analogen Voter gibt es zwei Möglichkeiten, dass dieser einen Mittelwert bildet, das bedeutet, dass dem Maschinenteil 26 die 1,355 übergeben wird. Ist der Voter so konfiguriert, dass er nur Werte übergeben soll, die in einem konfigurierbaren Bereich (z.B. zwischen 1,2 und 1,3) liegen, würde der Wert 1,23 dem Maschinenteil 26 übergeben. Die letzte Möglichkeit besteht darin, dass der Voter 3 „richtige" Werte (27-29) bekommt und dann der mittlere Werte übertragen wird.

[00198] Es wird nun ein Bereich zwischen 1,1 bis 1,3 definiert und entschieden, dass Werte in diesem Bereich den Ausgangswert 1,23 erhalten. Genau dieser Wert wird in der Übergabe 30 als Datenresultat 14-16 übernommen und schließlich an den daran anschließenden Maschinenteil 26 übergeben.

[00199] Selbstverständlich gibt es auch andere Möglichkeiten, an drei verschiedenen Eingängen 27-29 anliegende analoge Werte auf einen einzigen Wert zurückzuführen. Es kann eine Mittelwertbildung erfolgen oder auch andere statistische Rechenmethoden, die dazu führen, dass aus drei verschiedenen analogen Werten eine einzige, für diese analogen Werte repräsentative Wert gebildet wird.

[00200] Die Figur 12 zeigt nochmals in vereinfachter Form das Blockschaltbild nach den Figuren 1-3, wobei der Einfachheit halber lediglich eine einzige Peripherie-Einheit dargestellt ist und es ist erkennbar, dass gerade in dieser Peripherie-Einheit 7 die Werte der Applikation 1 abgearbeitet werden und dem daran anschließenden Maschinenteil 26 zugeführt werden.

[00201] Es handelt sich um ein klassisches IO-Device, welches lediglich Werte verarbeitet, die von der im Steuerrechner 1 oder Steuerrechner 2 laufenden Applikation 11 berechnet werden.

[00202] Die Figur 13 zeigt weitere Einzelheiten der Entscheidungsstufe 17-19, wie in Figur 3 und in den Figuren 10 und 11 dargestellt ist.

[00203] Hierbei ist erkennbar, dass an einem Eingang 31 die Werte P 1, P 2 und ferner die Werte S 1 und S 2 anliegen.

[00204] Aus diesen Werten muss eine Auswahl 32 getroffen werden und es ist erkennbar, dass aufgrund der Entscheidung der Transportschicht zunächst die Werte P 2 und S 2 an den Leitungen 35 und 37 nicht weiter berücksichtigt werden und gestrichen werden. Als nächstes wird entschieden, welcher der Werte auf der Leitung 34 oder 36 weiter berücksichtigt wird. Diese Entscheidung erfolgt im Voter 17-19, ob der Wert nun von dem Steuerrechner 1 oder von dem

Secondary 2 genommen wird oder wie oben beschrieben eine Mehrheitsentscheidung, Mittelwert durchgeführt werden soll.

[00205] Wie bereits vorhin erläutert sind der Wert 1 vom Primary-Rechner (P1) auf der Leitung 34 genau gleich wie der Wert 1 von dem Secondary-Rechner (S I) auf der Leitung 36 und im Voter 17-19 wird entschieden, ob nun der Wert auf der Leitung 34 vom Primary-Rechner oder der gleiche Wert auf der Leitung 36 vom Secondary-Rechner übernommen wird.

[00206] Beide Werte (S1 und P1) werden in die Entscheidungsstufe 17-19 eingespeist und in der Entscheidungsstufe wird - in Abhängigkeit von der Gesundheit der beiden Steuerrechner (siehe Abschnitt Fehler-Priorisierung II) - entschieden, ob der Wert P 1 vom Primary-Rechner oder der Wert S 1 vom Secondary-Rechner übernommen wird.

[00207] Auf Grund des besseren Gesundheitszustandes des Primary-Rechners wird also der Wert P 1 als Eingangswert in das Datenresultat 14 der Peripherie-Einheit eingespeist.

[00208] Dieser Wert steuert dann zum Beispiel einen Kanal Ch 1 mit dem Parameter P 1 an.

[00209] Somit kann im Maschinenteil 26 damit zum Beispiel eine Kontrollleuchte zum Ausleuchten gebracht werden.

[00210] Die Figur 14 zeigt in Erweiterung des Ausführungsbeispiels nach den Figuren 1 bis 3, dass auf den beiden Steuerrechnern 1, 3 eine beliebige Anzahl von genau zueinander gespiegelten Applikationen 11, 11', 12, 12' und 13, 13' laufen können, dass aber noch zusätzlich auch davon unabhängig nicht redundante Applikationen laufen können, für dies bei der Applikation 38 (App IV) der Fall ist, die zu keiner Applikation auf dem Steuerrechner 2 redundant ist.

[00211] Gleiches gilt als Beispiel für den Steuerrechner 2, wo eine weitere Applikation 39 (App V) unabhängig von allen anderen Applikationen ablaufen kann, wobei die beiden unabhängig voneinander ablaufenden und nicht redundanten Applikationen 38, 39 ebenfalls in gleicher Weise Werte über die Busleitung 5, 6 an die Peripherie-Einheiten senden können, und hierbei insbesondere die von den anderen Peripherie-Einheiten unabhängigen Peripherie-Einheiten 40, 41 mit Daten versorgen.

[00212] Die Erfindung ist also nicht auf die Anordnung von genau zueinander spiegelbildlich auf den beiden Steuerrechnern 1, 2 redundant ablaufenden Applikationen 11-13 beschränkt, sondern es können auch noch davon unabhängige Applikationen 38, 39 auf den beiden Steuerrechnern 1,2 ablaufen, um diese besser auszulasten.

ZEICHNUNGSLEGENDE

1 Steuerrechner 1
2 Steuerrechner 2
3 Busleitung (RI-Verbindung)
4 Busleitung (RI-Verbindung)
5 Busleitung (Arbeitsbus)
6 Busleitung (Arbeitsbus)
7 Peripherie-Einheit
8 Peripherie-Einheit
9 Peripherie-Einheit
10 Fabrik-Netzwerk
11 Applikation 11'
12 Applikation 12'
13 Applikation 13'
14 Datenresultat
15 Datenresultat

16 Datenresultat
17 Entscheidungsstufe
18 Entscheidungsstufe
19 Entscheidungsstufe
20 Eingangs- und Ausgangsleitung
21 Eingangs- und Ausgangsleitung
22 Eingangs- und Ausgangsleitung
23 Umschaltvorgang
24 Umschaltvorgang
25 Abgleichzeit
26 Maschinenteil
27 Eingang
28 Eingang
29 Eingang
30 Übergabe
31 Eingang (Werte)
32 Auswahl
33 Transportschicht
34 Leitung
35 Leitung
36 Leitung
37 Leitung
38 Applikation
39 Applikation
40 Peripherie-Einheit
41 Peripherie-Einheit
42 Umschaltimpuls

Ansprüche

1. Redundantes Steuerungssystem mit zumindest einem ersten Steuerrechner (1) und einem zweiten Steuerrechner (2), die miteinander über eine Redunanzkopplung (3, 4) verbunden sind, und zumindest einer der Steuerrechner (1, 2) zur Ausführung von Steuerungsaufgaben mit zumindest einer Peripherie-Einheit (7, 8, 9) über zumindest ein Bussystem (5, 6) Daten austauscht, **dadurch gekennzeichnet**, dass auf den mindestens zwei Steuerrechnern (1, 2) jeweils mehrere unterschiedliche Applikationen (11, 12, 13; 11', 12', 13') gleichzeitig und unabhängig voneinander ablaufen, dass die jeweilige Applikation (11, 12, 13) auf dem einen Steuerrechner (1) in identischer Form als gespiegelte Applikation (11', 12', 13') auf dem anderen Steuerrechner (2) abläuft und im Fehlerfall die von der fehlerhaften Applikation (11, 12, 13) angesteuerte Peripherie-Einheit (7-9) auf den anderen Steuerrechner und die dort noch lauffähige gleiche Applikation (11', 12', 13') umschaltet.

2. Steuerungssystem nach Anspruch 1, **dadurch gekennzeichnet**, dass jede der Applikationen (11, 11'; 12, 12', 13, 13') gesondert unter eigenem Namen adressierbar ist und entsprechend modifizierbar, löschbar oder installierbar ist, ohne dass die anderen Applikationen (PLC-Projekte) beeinflusst werden.

3. Steuerungssystem nach einem der Ansprüche 1 oder 2, **dadurch gekennzeichnet**, dass die Synchronisation der redundanten Applikationen (11, 11'; 12, 12', 13, 13') im Zyklus der jeweiligen Applikation (11, 11'; 12, 12', 13, 13') erfolgt.

4. Steuerungssystem nach einem der Ansprüche 1 bis 3, **dadurch gekennzeichnet**, dass die Master-CPUs in den zueinander redundanten Steuerrechnern (1, 2) synchronisiert werden

5. Steuerungssystem nach einem der Ansprüche 1 bis 4, **dadurch gekennzeichnet**, dass der Datenaustausch zu der Peripherie (7-9) mit einer Umschaltung in Echtzeit geschieht.

6. Verfahren zum Betrieb eines redundanten Steuerungssystems mit zumindest einem ersten Steuerrechner (1) und einem zweiten Steuerrechner (2), die miteinander über eine Redunanzkopplung (3, 4) verbunden sind, und zumindest einer der Steuerrechner (1, 2) zur Ausführung von Steuerungsaufgaben mit zumindest einer Peripherie-Einheit (7, 8, 9) über zumindest ein Bussystem (5, 6) Daten austauscht, **dadurch gekennzeichnet**, dass auf den mindestens zwei Steuerrechnern (1, 2) jeweils mehrere unterschiedliche Applikationen (11, 12, 13; 11', 12', 13') gleichzeitig und unabhängig voneinander ablaufen, dass die jeweilige Applikation (11, 12, 13) auf dem einen Steuerrechner (1) in identischer Form als gespiegelte Applikation (11', 12', 13') auf dem anderen Steuerrechner (2) abläuft und im Fehlerfall die von der fehlerhaften Applikation (11, 12, 13) angesteuerte Peripherie-Einheit (7-9) auf den anderen Steuerrechner und die dort noch lauffähige gleiche Applikation (11', 12', 13') umschaltet.

7. Verfahren zum Betrieb eines redundanten Steuerungssystems nach Anspruch 6, **dadurch gekennzeichnet**, dass auf den mindestens zwei Steuerrechnern (1, 2) eine Bewertung (Gewichtung) des Fehlerzustandes des jeweiligen Steuerrechners (1, 2) erfolgt, und dass in einer Vergleichseinheit die beiden von den Steuerrechnern (1, 2) ermittelten Fehler gegeneinander abgewogen werden, und in Abhängigkeit von dieser Abwägung vom einen auf den anderen Steuerrechner (1, 2) umgeschaltet wird.

8. Verfahren zum Betrieb eines redundanten Steuerungssystems nach Anspruch 6 oder 7, **dadurch gekennzeichnet,** dass die Umschaltung vom einen Steuerrechner (1, 2) auf den anderen Steuerrechner (2, 1) immer von dem Steuerrechner (1, 2) ausgelöst wird, der als gegenwärtige primary CPU mit der Peripherie-Einheit (7-9) Daten austauscht.

9. Verfahren zum Betrieb eines redundanten Steuerungssystems nach einem oder mehreren der vorhergehenden Ansprüche 6 bis 8, **dadurch gekennzeichnet**, dass in jedem Zyklus auf beiden CPUs der Fehler mit dem größten FailLevel und der zugehörigen Auftretenshäufigkeit ermittelt wird, dass diese Information im zyklischen Frame an die jeweils andere CPU übertragen wird und daraus die Primary-CPU ermittelt, ob sich die Secondary-CPU in einem besseren Zustand befindet und ob im Fehlerfall ein Failover ausgelöst werden soll.

10. Verfahren zum Betrieb eines redundanten Steuerungssystems nach einem oder mehreren der vorhergehenden Ansprüche 6 bis 9, **dadurch gekennzeichnet**, dass die benötigte Software (Betriebssystem, Treiber, Dienste, Applikation usw.) der Master-CPUs der beiden Steuerrechner (1, 2) und die Prozessvariablen der redundanten Applikationen (11, 12, 13; 11', 12', 13') zwischen den Master-CPUs der zueinander redundanten Steuerrechner (1, 2) abgeglichen werden.

11. Verfahren zum Betrieb eines redundanten Steuerungssystems nach Anspruch 10, **dadurch gekennzeichnet**, dass der Abgleich der Prozessvariablen zyklisch stattfindet und der Abgleich der Software beim Boot-Vorgang durchgeführt wird.

12. Verfahren zum Betrieb eines redundanten Steuerungssystems nach einem oder mehreren der vorhergehenden Ansprüche 6 bis 11, **dadurch gekennzeichnet**, dass die jeweilige Applikation (11, 12, 13) auf der Primary-CPU und die hierzu redundante Applikation (11', 12', 13') auf der Secondary-CPU ihren jeweilig als redundant gehaltenen Speicher anmelden, der in Blöcke (z.B. max. Größe 1400Bytes) unterteilt und zur anderen CPU übermittelt wird, dass in einem Übertragungszyklus zunächst der Block mit Variablen (V7-V9) übertragen wird, die sich während eines Zyklus nicht oder nur wenig verändert haben, dass danach der Block mit Variablen (V4-V6), die sich während des Zyklus stärker verändert haben und dass schließlich der Block mit Variablen (V1-V3) übertragen wird, die sich am stärksten verändert haben.

13. Verfahren zum Betrieb eines redundanten Steuerungssystems nach einem oder mehreren der vorhergehenden Ansprüche 6 bis 12, **dadurch gekennzeichnet**, dass grundsätzlich die Werte von der Primary-CPU an die Peripherie-Einheit (7-9) weitergeleitet werden, wenn jedoch die Peripherie-Einheit (7-9) keine Daten von der Primary-CPU bekommt, werden die Werte von der Secondary-CPU an die Peripherie-Einheit (7-9) weitergeleitet.

14. Verfahren zum Betrieb eines redundanten Steuerungssystems nach Anspruch 13, **dadurch gekennzeichnet**, dass eine automatische Umschaltung nach einer vorgegebenen Anzahl von Zyklen erfolgt.

15. Verfahren zum Betrieb eines redundanten Steuerungssystems nach Anspruch 13 und/oder 14, **dadurch gekennzeichnet**, dass aus den synchronen Daten der beiden Master-CPU ein Mittelwert gebildet wird, der zur jeweiligen Applikation (11, 12, 13; 11', 12', 13') weitergeleitet wird.

16. Verfahren zum Betrieb eines redundanten Steuerungssystems nach einem oder mehreren der vorhergehenden Ansprüche 6 bis 15, **dadurch gekennzeichnet**, dass bei Vorliegen von drei digitalen Werten an jeder Master-CPU jedes Steuerrechners (1, 2) eine Mehrheitsentscheidung darüber getroffen wird, welcher Wert an die Peripherie-Einheit (7-9) übermittelt wird.

17. Verfahren zum Betrieb eines redundanten Steuerungssystems nach einem oder mehreren der vorhergehenden Ansprüche 6 bis 16, **dadurch gekennzeichnet**, dass bei Vorliegen von mehreren analogen Werten an jeder Master-CPU jedes Steuerrechners (1, 2) ein Mittelwert aus allen analogen Werten gebildet wird, und dieser Mittelwert an die Peripherie-Einheit (7-9) übermittelt wird.

18. Verfahren zum Betrieb eines redundanten Steuerungssystems nach einem oder mehreren der vorhergehenden Ansprüche 6 bis 17, **dadurch gekennzeichnet**, dass redundante und nicht redundante Applikationen auf einer Master-CPU gleichzeitig lauffähig sind.

Hierzu 8 Blatt Zeichnungen

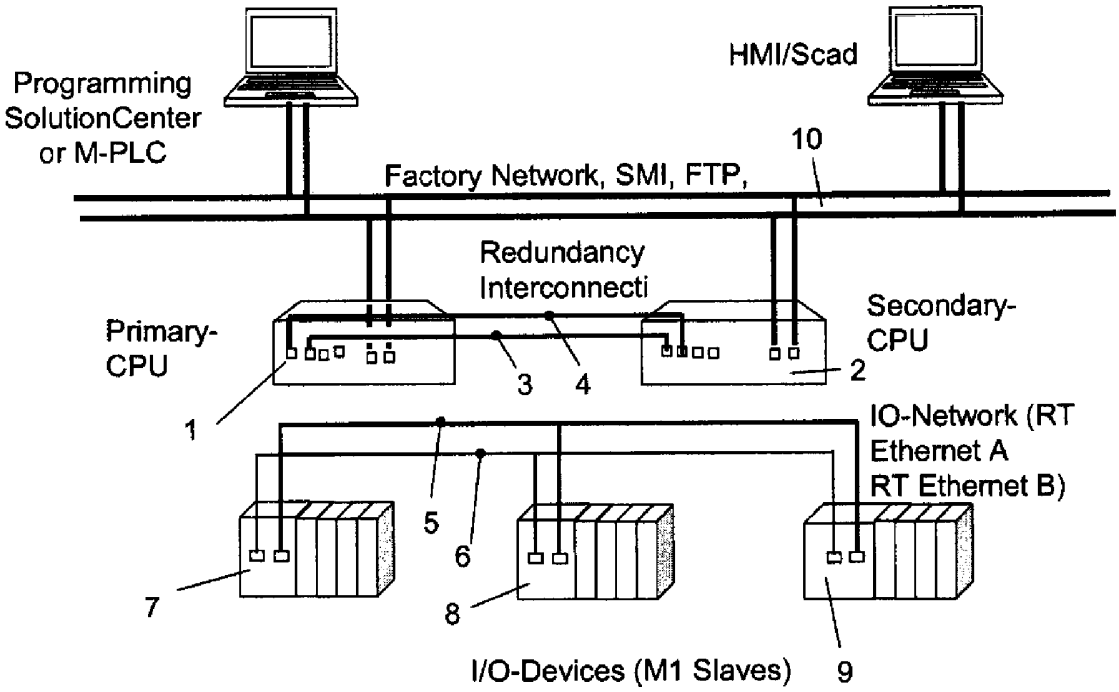

Programming
SolutionCenter
or M-PLC
HMI/Scad
10
Factory Network, SMI, FTP,
Redundancy
Interconnecti
Primary-
CPU
Secondary-
CPU
1
2
3
4
IO-Network (RT
Ethernet A
RT Ethernet B)
5
6
7
8
I/O-Devices (M1 Slaves)
9

Fig. 1

Fig. 2

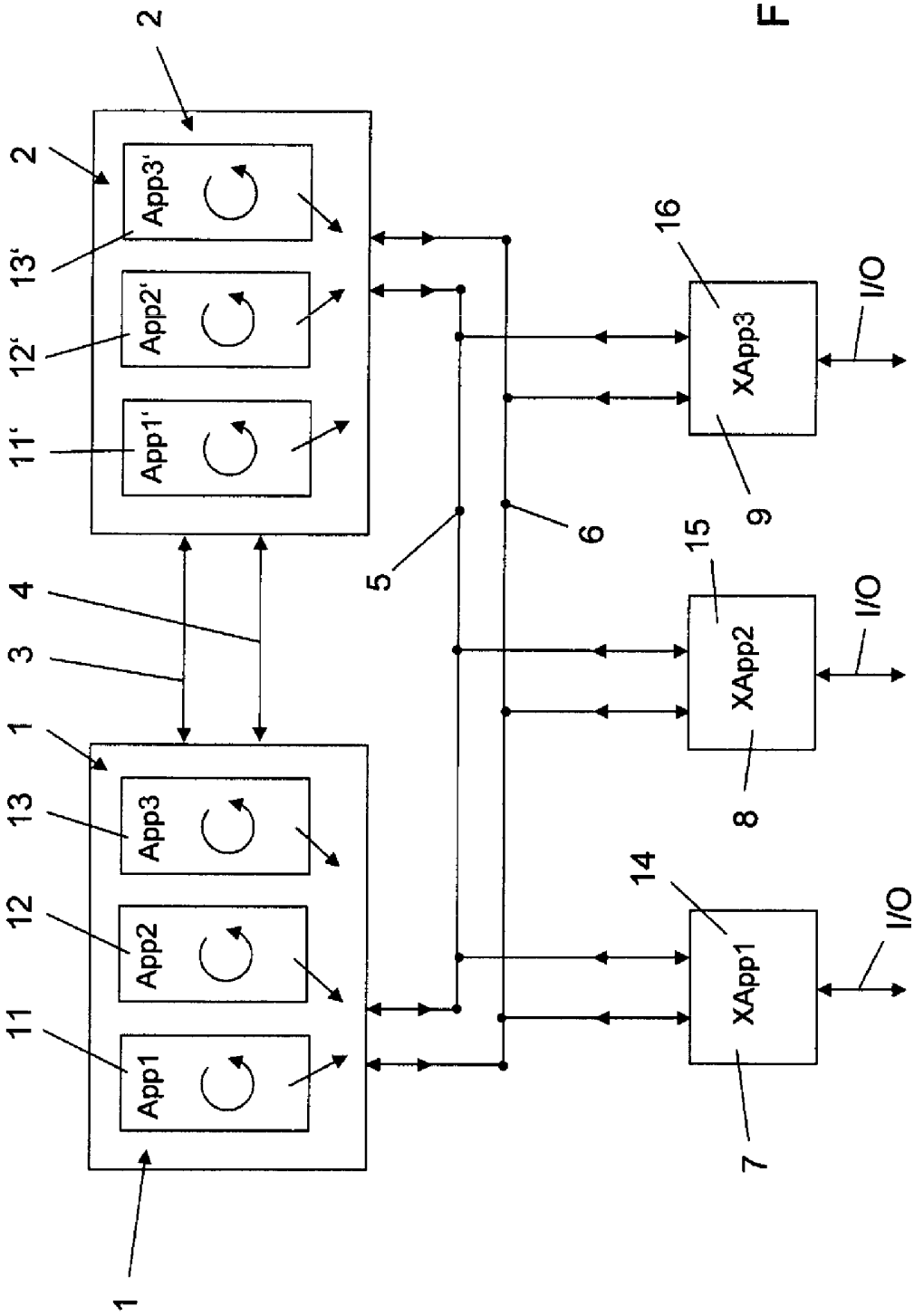
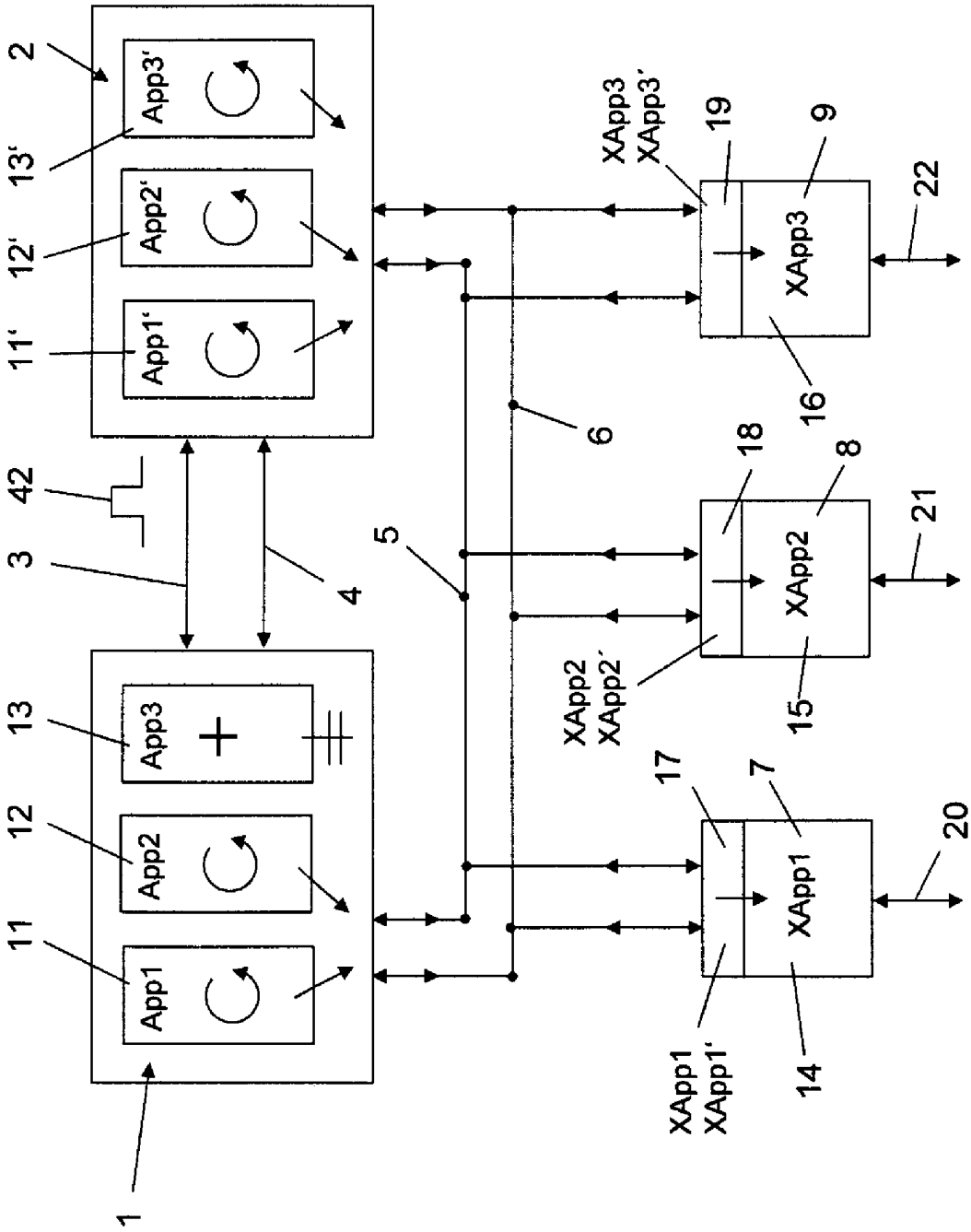

2
2
13'
12'
11'
App3'
App2'
App1'
16
XApp3
I/O
9
5
6
4
3
15
XApp2
I/O
8
1
13
12
11
App3
App2
App1
14
XApp1
I/O
7
1

Fig. 3


1
2
App3'
13'
App2'
12'
App1'
11'
42
3
4
5
6
App3
13
App2
12
App1
11
XApp3
XApp3'
19
9
22
16
18
8
21
XApp2
XApp2'
15
17
7
20
XApp1
14
XApp1'

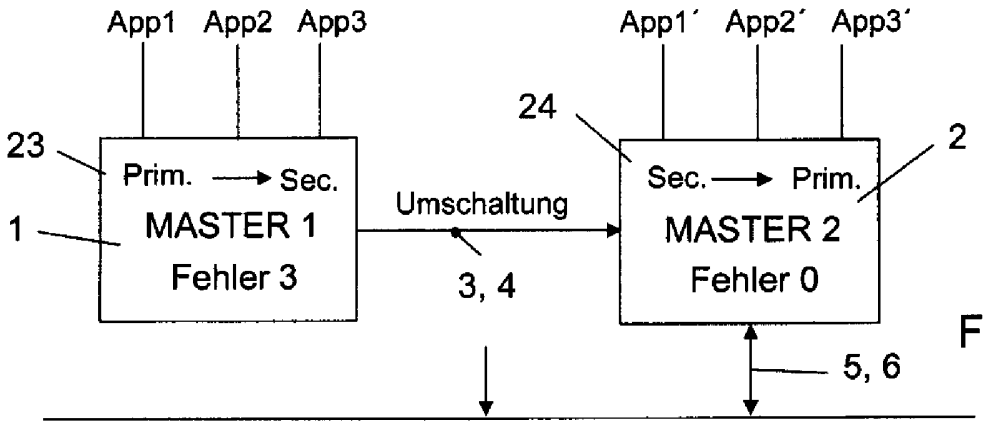


Fig. 4

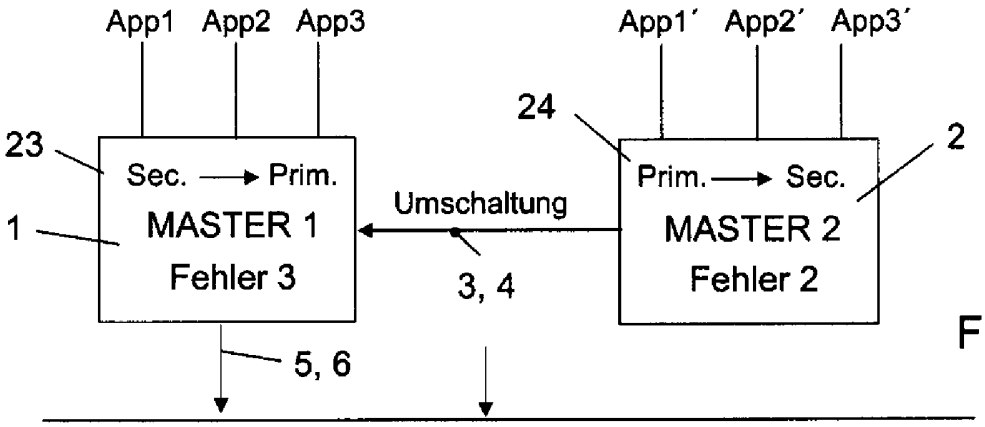


Fig. 5

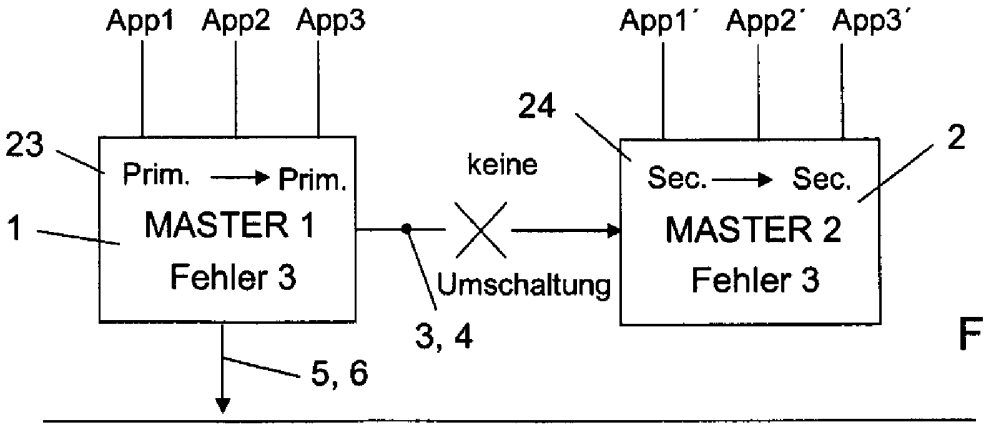


Fig. 6

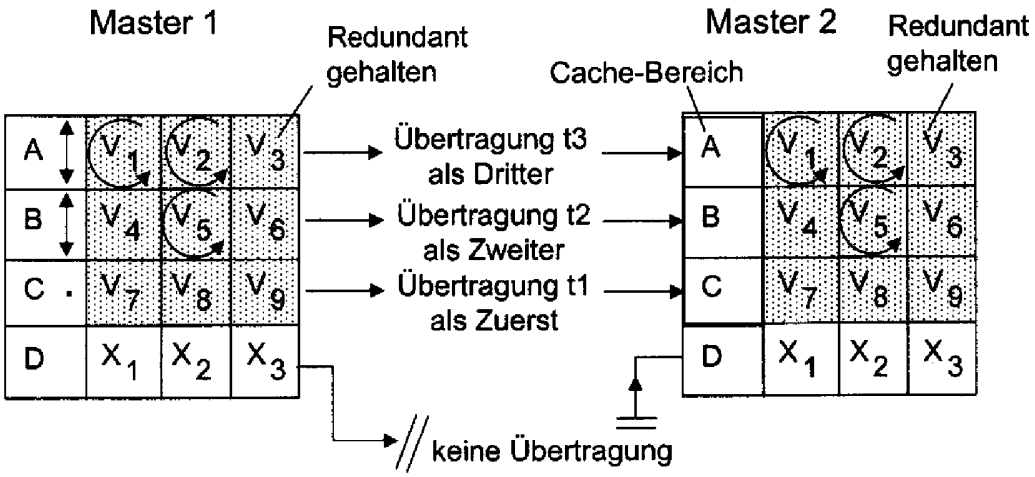


Fig. 7

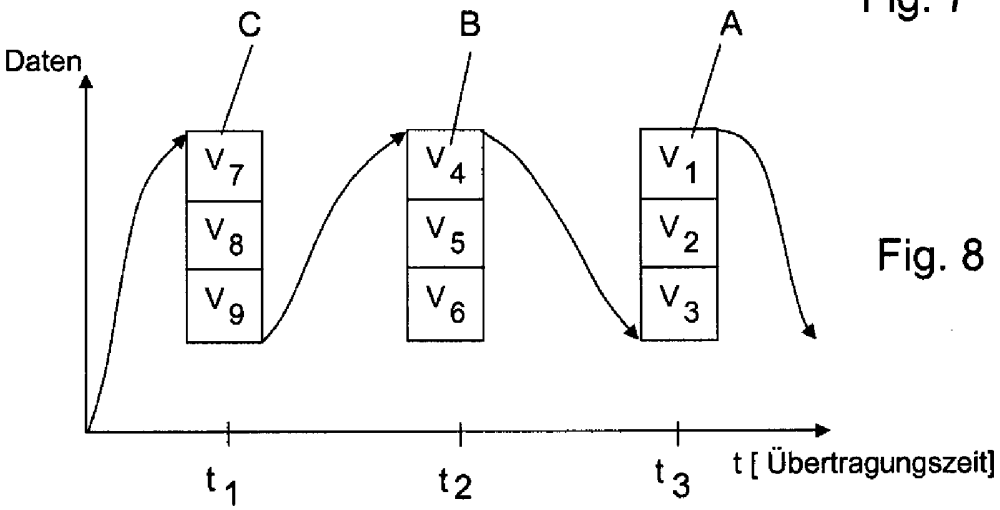


Fig. 8

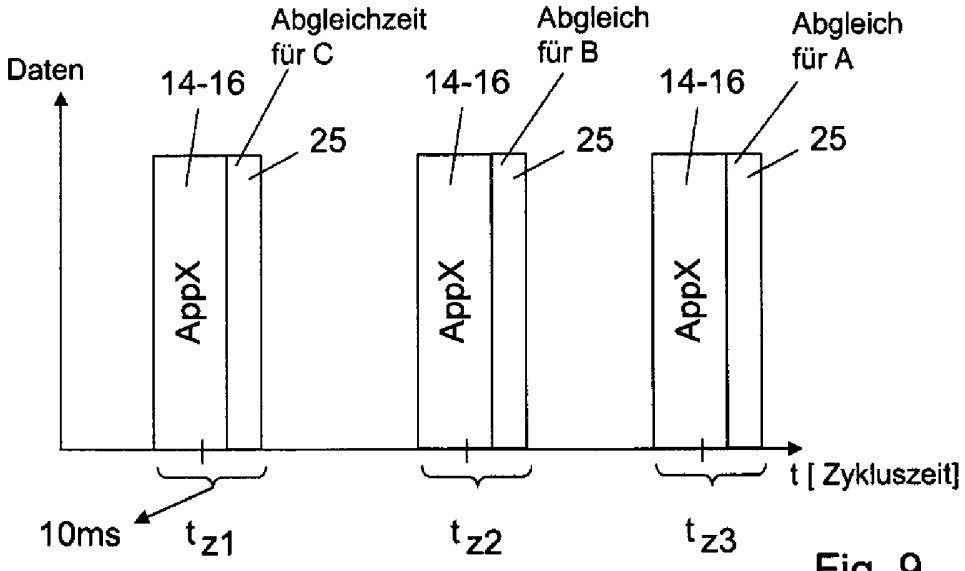


Fig. 9

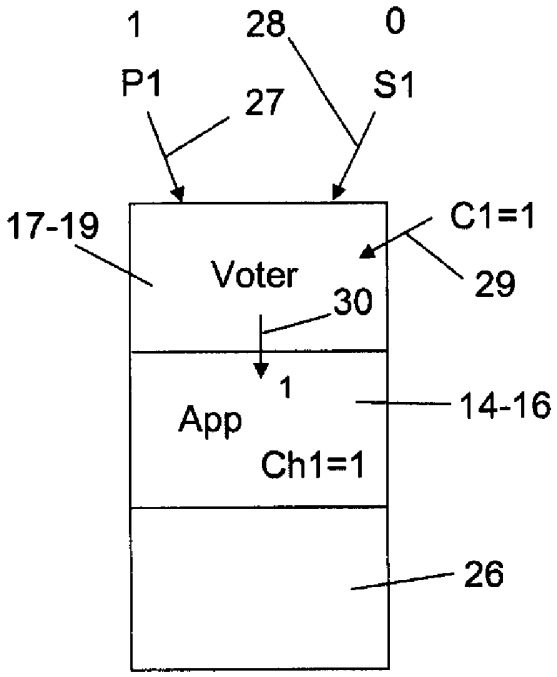

1
28
0
P1
27
S1
17-19
C1=1
Voter
29
30
1
App
14-16
Ch1=1
26

Fig. 10

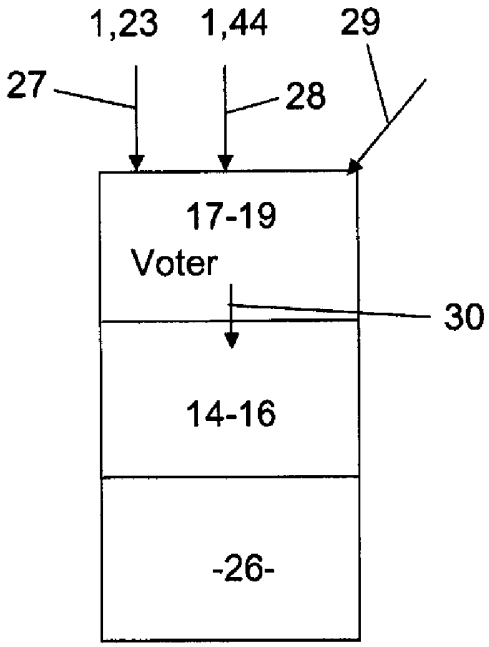

1,23
1,44
29
27
28
17-19
Voter
30
14-16
-26-

Fig. 11

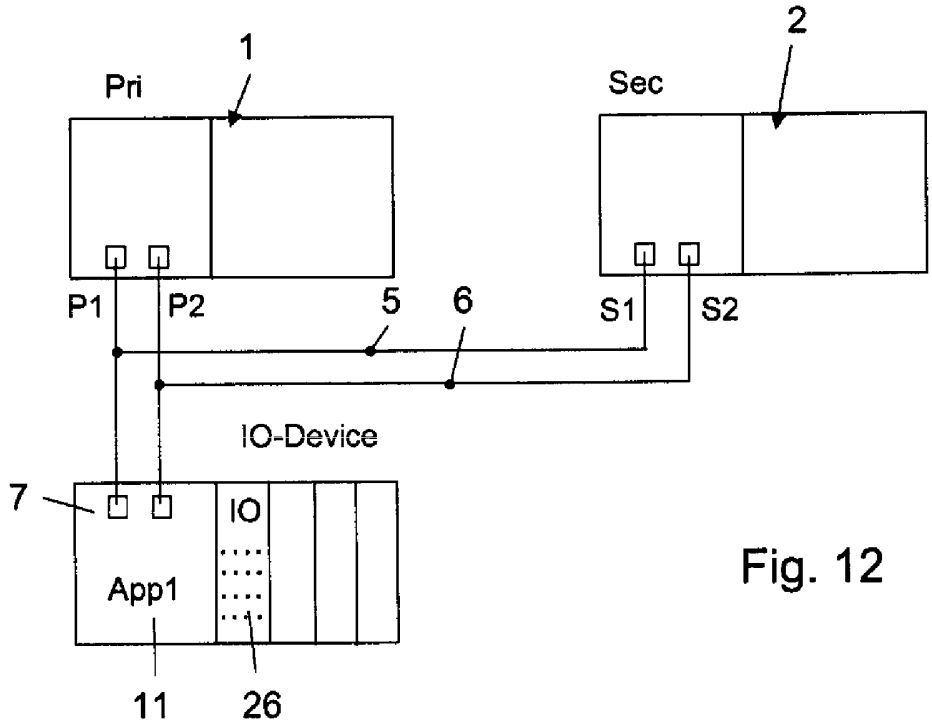

1
Pri
2
Sec
P1
P2
5
6
S1
S2
IO-Device
7
IO
App1
11
26

Fig. 12

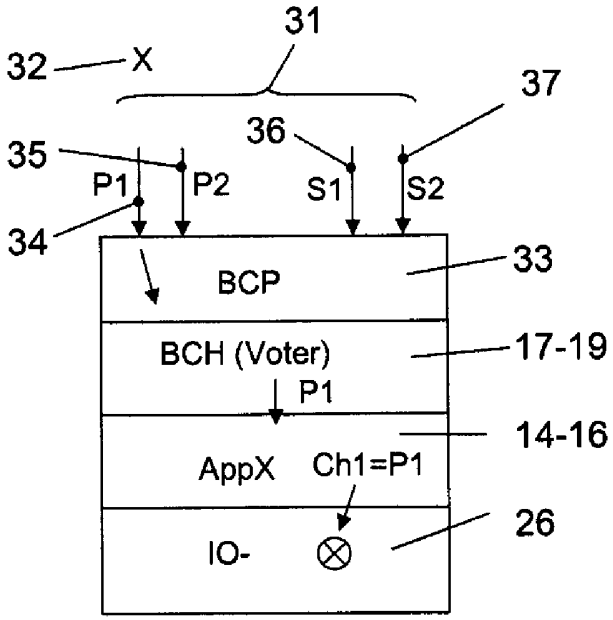

31
32
X
37
36
35
P1
P2
S1
S2
34
33
BCP
BCH (Voter)
17-19
P1
14-16
AppX
Ch1=P1
26
IO-

Fig. 13

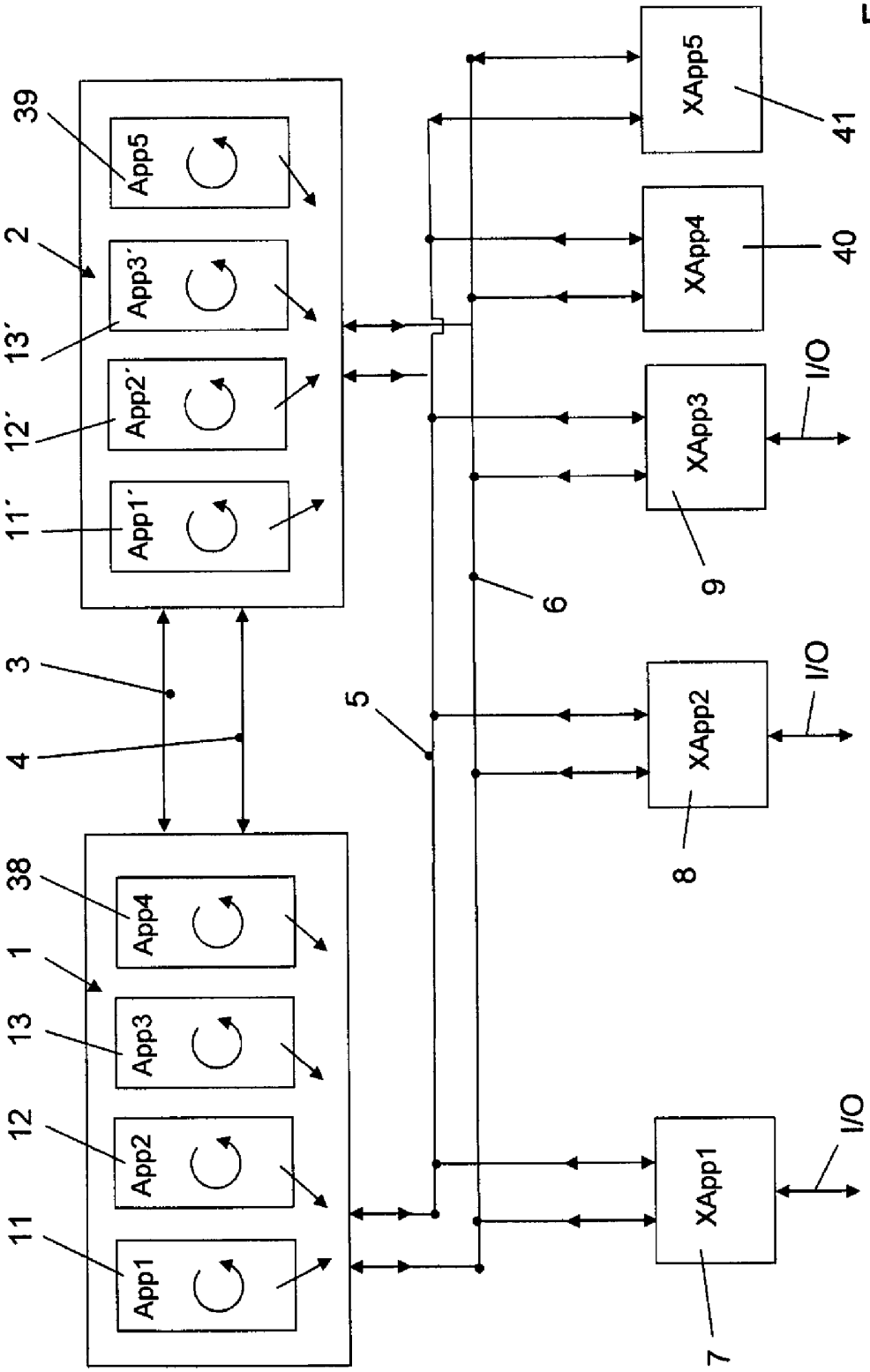

App1
App2
App3
App4
App1'
App2'
App3'
App5
XApp1
XApp2
XApp3
XApp4
XApp5
I/O
I/O
I/O
11
12
13
38
1
11'
12'
13'
2
39
3
4
5
6
7
8
9
40
41

Fig. 14

Recherchenbericht zu **GM 8/2012**

österreichisches patentamt

Klassifikation des Anmeldungsgegenstands gemäß IPC: ***G05B 19/05*** (2006.01); ***G06F 11/16*** (2006.01)
Klassifikation des Anmeldungsgegenstands gemäß ECLA: G05B 19/05S; G06F 11/16C8; G06F 11/16C12
Recherchierter Prüfstoff (Klassifikation): G05B, G06F
Konsultierte Online-Datenbank: EPODOC, WPI, TXTDE, TXTEN
Dieser Recherchenbericht wurde zu den **am 14. August 2012 eingereichten** Ansprüchen **1-18** erstellt. Die in der Gebrauchsmusterschrift veröffentlichten Ansprüche könnten im Verfahren geändert worden sein (§ 19 Abs. 4 GMG), sodass die Angaben im Recherchenbericht, wie Bezugnahme auf bestimmte Ansprüche, Angabe von Kategorien (X, Y, A), nicht mehr zutreffend sein müssen. In die dem Recherchenbericht zugrundeliegende Fassung der Ansprüche kann beim Österreichischen Patentamt während der Amtsstunden Einsicht genommen werden.

Kategorie*)	Bezeichnung der Veröffentlichung: Ländercode, Veröffentlichungsnummer, Dokumentart (Anmelder), Veröffentlichungsdatum, Textstelle oder Figur soweit erforderlich	Betreffend Anspruch
X	US 2008125886 A1 (KING ET AL.) 29. Mai 2008 (29.05.2008) Zusammenfassung; Figur; Beschreibung der Figur; Ansprüche 1-2, 5, 7, 10, 13-17;	1-6, 8, 10-11, 13-14, 18
A		7, 9, 12, 15-17
A	DE 102009050449 B3 (SIEMENS AKTIENGESELLSCHAFT) 09. Dezember 2010 (09.12.2010) Zusammenfassung; Figuren 1-2; Beschreibung der Figuren; Ansprüche 1-2, 4-10;	1-18
A	WO 2005052703 A1 (SIEMENS AKTIENGESELLSCHAFT) 09. Juni 2005 (09.06.2005) Zusammenfassung; Figur; Beschreibung der Figur; Ansprüche 1-8;	1-18

Datum der Beendigung der Recherche:	☐ Fortsetzung siehe Folgeblatt	Prüfer(in):
1. Oktober 2012		STOLL J.

*) **Kategorien** der angeführten Dokumente:

- **X** Veröffentlichung **von besonderer Bedeutung**: der Anmeldungsgegenstand kann allein aufgrund dieser Druckschrift nicht als neu bzw. auf erfinderischer Tätigkeit beruhend betrachtet werden.
- **Y** Veröffentlichung **von Bedeutung**: der Anmeldungsgegenstand kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren weiteren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese **Verbindung für einen Fachmann naheliegend** ist.
- **A** Veröffentlichung, die den **allgemeinen Stand der Technik** definiert.
- **P** Dokument, das **von Bedeutung** ist (Kategorien **X** oder **Y**), jedoch **nach dem Prioritätstag** der Anmeldung **veröffentlicht** wurde.
- **E** Dokument, das **von besonderer Bedeutung** ist (Kategorie **X**), aus dem ein **älteres Recht** hervorgehen könnte (früheres Anmeldedatum, jedoch nachveröffentlicht, Schutz ist in Österreich möglich, würde Neuheit in Frage stellen).
- **&** Veröffentlichung, die Mitglied der selben **Patentfamilie** ist.