(19) Deutsches Patent- und Markenamt

(10) **DE 10 2020 208 367 A1** 2021.01.21

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2020 208 367.3**
(22) Anmeldetag: **03.07.2020**
(43) Offenlegungstag: **21.01.2021**

(51) Int Cl.: ***G06F 9/52*** (2006.01)

(30) Unionspriorität:
2019-132040 17.07.2019 JP

(71) Anmelder:
DENSO CORPORATION, Kariya-city, Aichi-pref., JP

(74) Vertreter:
Winter, Brandl, Fürniss, Hübner, Röss, Kaiser, Polte Partnerschaft mbB, Patentanwälte, 85354 Freising, DE

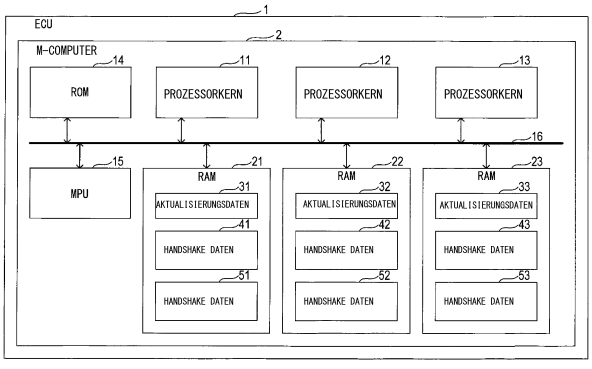
(72) Erfinder:
Kunibe, Hirotaka, Kariya-city, Aichi-pref., JP

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **ELEKTRONISCHE STEUEREINHEIT**

(57) Zusammenfassung: Eine ECU (1) enthält eine Vielzahl von Prozessorkernen (11, 12, 13), ein RAM (21) und eine MPU (15). Der RAM (21) speichert Aktualisierungsdaten (31), auf die von jedem der Prozessorkerne (11, 12, 13) lesbar zugegriffen werden kann. Die MPU (15) steuert die Prozessorkerne (11, 12, 13), so dass, wenn ein spezifischer Prozessorkern (11) auf die Aktualisierungsdaten (31) zugreift, der/die Prozessorkern(e) (12, 13) mit Ausnahme des spezifischen Prozessorkerns nicht auf die Aktualisierungsdaten (31) zugreifen kann/können.



Beschreibung

TECHNISCHER BEREICH

[0001] Die vorliegende Offenbarung bezieht sich auf eine elektronische Steuereinheit mit einer Vielzahl von Prozessorkernen.

HINTERGRUND

[0002] In Patentliteratur 1 wird eine fahrzeuginterne Steuereinheit beschrieben, die einen ersten Kern, einen zweiten Kern und einen gemeinsamen Speicher umfasst. In der fahrzeuginternen Steuereinheit werden die Daten des zweiten Kerns unter Verwendung einer Hardware-Funktion wie DMA direkt in den gemeinsamen Speicher kopiert; der erste Kern greift auf den gemeinsamen Speicher zu und erfasst die Daten. DMA ist eine Abkürzung für Direct Memory Access (direkter Speicherzugriff).

[0003] Patentliteratur 1: WO 2017/056725 A1

KURZZUSAMMENFASSUNG

[0004] Die ausschließliche Steuerung zwischen den Prozessorkernen erfolgt in der Regel über eine Semaphore. Wenn ein Semaphor verwendet wird, muss ein zugreifender Prozessorkern immer eine Schnittstelle für den Erwerb von Semaphoren oder die Freigabe von Semaphoren verwenden, was zu einer Erhöhung des Overheads führt.

[0005] Gegenstand der vorliegenden Offenbarung ist es, eine Erhöhung des Overheads aufgrund der ausschließlichen Steuerung zwischen Prozessorkernen zu unterdrücken.

[0006] Nach einem Beispiel der vorliegenden Offenbarung ist eine elektronische Steuereinheit vorgesehen, die eine Vielzahl von Prozessorkernen, eine gemeinsam genutzte Datenspeichereinheit und eine Speicherschutzeinheit umfasst. Die gemeinsam genutzte Datenspeichereinheit ist so konfiguriert, dass sie gemeinsam genutzte Daten speichert, auf die von jedem der Vielzahl von Prozessorkernen lesbar zugegriffen werden kann.

[0007] Die Speicherschutzeinheit ist so konfiguriert, dass sie die Vielzahl der Prozessorkerne steuert, so dass, wenn ein spezifischer Prozessorkern auf die gemeinsam genutzten Daten zugreift, ein anderer Prozessorkern nicht auf die gemeinsam genutzten Daten zugreifen kann. Der spezifische Prozessorkern ist einer der Vielzahl von Prozessorkernen. Der andere Prozessorkern ist mindestens einer der Vielzahl von Prozessorkernen mit Ausnahme des spezifischen Prozessorkerns.

[0008] Wenn in der elektronischen Steuereinheit gemäß der vorliegenden Offenbarung der spezifische Prozessorkern auf die gemeinsam genutzten Daten zugreift, ist die Speicherschutzeinheit so konfiguriert, dass sie die Vielzahl der Prozessoren bzw. Prozessorkerne so steuert, dass der andere Prozessorkern nicht auf die gemeinsam genutzten Daten zugreifen kann. Das bedeutet, dass die elektronische Steuereinheit der vorliegenden Offenbarung die Speicherschutzeinheit anstelle einer Semaphore verwendet; daher muss der Prozessorkern, auf den zugegriffen wird, keine spezielle Schnittstelle enthalten. Dementsprechend kann die elektronische Steuereinheit der vorliegenden Offenbarung eine Erhöhung des Overheads aufgrund der exklusiven Steuerung zwischen den Prozessorkernen unterdrücken.

Figurenliste

[0009] Die oben genannten und anderen Aufgaben, Merkmale und Vorteile der vorliegenden Offenbarung werden in der folgenden detaillierten Beschreibung, die unter Bezugnahme auf die beigefügten Zeichnungen erstellt wurde, deutlicher hervortreten. In den Zeichnungen:

Fig. 1 ist ein Blockdiagramm, das eine Konfiguration eines Steuergeräts nach der ersten und zweiten Ausführungsform veranschaulicht;

Fig. 2 ist ein Diagramm, das eine Konfiguration einer Prüfblock-ID-Tabelle veranschaulicht;

Fig. 3 ist ein Flussdiagramm, das einen anfänglichen Einstellprozess veranschaulicht;

Fig. 4 ist ein Flussdiagramm, das einen RAM-Testprozess veranschaulicht;

Fig. 5 ist ein Flussdiagramm, das einen Zugriffsfehlerprozess veranschaulicht;

Fig. 6 ist ein Sequenzdiagramm, das ein erstes konkretes Beispiel für eine Zugriffsentscheidung durch eine MPU nach der ersten Ausführungsform illustriert;

Fig. 7 ist ein Sequenzdiagramm, das ein zweites spezifisches Beispiel für eine Zugriffsentscheidung durch die MPU nach der ersten Ausführungsform veranschaulicht;

Fig. 8 ist ein Diagramm, das eine ausschließliche Steuerung durch ein Semaphor illustriert;

Fig. 9 ist ein Diagramm, das eine ausschließliche Steuerung durch einen Mikrocomputer nach der ersten Ausführungsform veranschaulicht;

Fig. 10 ist ein Diagramm, das eine Konfiguration einer Prioritätstabelle nach einer zweiten Ausführungsform illustriert;

Fig. 11 ist ein Flussdiagramm, das einen ersten Prozessorprozess gemäß der zweiten Ausführungsform veranschaulicht;

Fig. 12 ist ein Flussdiagramm, das einen Datenleseprozess gemäß der zweiten Ausführungsform illustriert;

Fig. 13 ist ein Sequenzdiagramm, das ein spezifisches Beispiel für eine Zugriffsentscheidung durch eine MPU nach der zweiten Ausführungsform illustriert;

Fig. 14 ist ein Blockdiagramm, das eine Konfiguration eines Steuergeräts nach der dritten und vierten Ausführungsform veranschaulicht;

Fig. 15 ist ein Sequenzdiagramm, das ein erstes spezifisches Beispiel für eine Ausführungszugriffsentscheidung durch eine MPU gemäß der dritten Ausführungsform veranschaulicht;

Fig. 16 ist ein Sequenzdiagramm, das ein zweites spezifisches Beispiel einer Ausführungszugriffsentscheidung durch die MPU gemäß der dritten Ausführungsform veranschaulicht; und

Fig. 17 ist ein Sequenzdiagramm, das ein spezifisches Beispiel für eine Ausführungszugriffsentscheidung durch die MPU nach einer vierten Ausführungsform veranschaulicht.

DETAILLIERTE BESCHREIBUNG

[Erste Ausführungsform]

[0010] Im Folgenden wird eine erste Ausführungsform der vorliegenden Offenbarung mit Bezug auf die Zeichnungen beschrieben. Wie in **Fig. 1** dargestellt, enthält eine elektronische Steuereinheit **1** (im Folgenden ECU **1**) der vorliegenden Ausführungsform einen Mikrocomputer **2**. ECU ist eine Abkürzung für Electronic Control Unit (elektronische Steuereinheit).

[0011] Der Mikrocomputer **2** umfasst die Prozessorkerne **11**, **12** und **13**, ein ROM **14**, eine Speicherschutzeinheit **15** (nachfolgend MPU **15**), einen Systembus **16** und die RAMs **21**, **22** und **23**. MPU ist eine Abkürzung für Memory Protection Unit (Speicherschutzeinheit).

[0012] Verschiedene Funktionen des Mikrocomputers **2** werden durch die Prozessorkerne **11**, **12** und **13** realisiert, die Programme ausführen, die in einem nicht vorübergehenden, greifbaren Speichermedium gespeichert sind. In diesem Beispiel entspricht das ROM **14** einem nicht transitorischen, greifbaren Speichermedium, das ein Programm speichert. Darüber hinaus wird durch die Ausführung dieses Programms eine dem Programm entsprechende Methode ausgeführt. Beachten Sie, dass einige oder alle Funktionen, die von den Prozessorkernen **11**, **12** und **13** ausgeführt werden, von einem oder mehreren ICs o.ä. als Hardware konfiguriert werden können. Ferner kann die Anzahl der Mikrocomputer, aus denen die ECU **1** besteht, ein oder mehrere sein.

[0013] Die Prozessorkerne **11**, **12** und **13** enthalten jeweils eine Recheneinheit und ein Register zur Ausführung eines Programms. Die Prozessorkerne **11**, **12** und **13** führen verschiedene Steuerprozesse zur Steuerung eines Motors (nicht abgebildet) verteilt aus.

[0014] Das ROM **14** ist ein nichtflüchtiger Speicher, in den Daten nicht überschrieben werden können. Das ROM **14** speichert Programme, die von den Prozessorkernen **11**, **12** und **13** ausgeführt werden. Die MPU **15** steuert das Schreiben und Lesen von Daten in und aus den RAMs **21**, **22** und **23**.

[0015] Der Systembus **16** verbindet die Prozessorkerne **11**, **12** und **13**, das ROM **14**, die MPU **15** und die RAMs **21**, **22** und **23** miteinander, so dass Daten ein- und ausgegeben werden können.
Die RAMs **21**, **22** und **23** sind flüchtige Speicher. Die RAMs **21**, **22** und **23** speichern vorübergehend Berechnungsergebnisse der Prozessorkerne **11**, **12** bzw. **13**.

[0016] Der RAM **21** speichert die Aktualisierungsdaten **31** und die Handshake-Daten **41** und **51**. Die Aktualisierungsdaten **31** sind Daten, die ausschließlich durch den Prozessorkern **11** aktualisiert werden. Die Handshake-Daten **41** sind die Daten, die verwendet werden, wenn der Prozessorkern **12** auf die Aktualisierungsdaten **31** zugreift. Bei den Handshake-Daten **51** handelt es sich um Daten, die verwendet werden, wenn der Prozessorkern **13** auf die Aktualisierungsdaten **31** zugreift.

[0017] Der RAM **22** speichert die Aktualisierungsdaten **32** und die Handshake-Daten **42** und **52**. Die Aktualisierungsdaten **32** sind Daten, die ausschließlich durch den Prozessorkern **12** aktualisiert werden. Bei den Handshake-Daten **42** handelt es sich um Daten, die verwendet werden, wenn der Prozessorkern **11** auf die Aktualisierungsdaten **32** zugreift. Die Handshakedaten **52** sind Daten, die verwendet werden, wenn der Prozessorkern **13** auf die Aktualisierungsdaten **32** zugreift.

[0018] Der RAM **23** speichert die Aktualisierungsdaten **33** und die Handshake-Daten **43** und **53**. Die Aktualisierungsdaten **33** sind Daten, die ausschließlich durch den Prozessorkern **13** aktualisiert werden. Die Handshake-Daten **43** sind Daten, die verwendet werden, wenn der Prozessorkern **11** auf die Aktualisierungsdaten **33** zugreift. Bei den Handshake-Daten **53** handelt es sich um Daten, die verwendet werden, wenn der Prozessorkern **12** auf die Aktualisierungsdaten **33** zugreift.

[0019] Die MPU **15** steuert die Prozessorkerne **11**, **12** und **13**, so dass nur der Prozessorkern **11** die Aktualisierungsdaten **31** im RAM **21** aktualisieren kann. In ähnlicher Weise steuert die MPU **15** die Prozessor-

kerne **11**, **12** und **13**, so dass nur der Prozessorkern **12** die Aktualisierungsdaten **32** im RAM **22** aktualisieren kann. Darüber hinaus steuert die MPU **15** die Prozessorkerne **11**, **12** und **13**, so dass nur der Prozessorkern **13** die Aktualisierungsdaten **33** im RAM **23** aktualisieren kann.

[0020] Wenn außerdem eine Lesezugriffsanforderung auf die RAMs **21**, **22** und **23** von den Prozessorkernen **11**, **12** und **13** eingegeben wird, entscheidet die MPU **15**, ob ein Lesezugriff erlaubt oder verboten werden soll. Dann gibt die MPU **15** (i) eine Lesezugriffserlaubnismeldung an die Quelle der Zugriffsanforderung aus, wenn der Lesezugriff erlaubt ist, und (ii) eine Lesezugriffsverbotsmeldung, wenn der Lesezugriff verboten ist.

[0021] Die Prozessorkerne **11**, **12** und **13** bestimmen, ob ein Lesezugriff auf der Grundlage der Lesezugriffsberechtigungsbenachrichtigung oder der Lesezugriffsverbotsbenachrichtigungseingabe von der MPU **15** ausgeführt werden soll.

[0022] Das ROM **14** speichert eine Testblock-ID-Tabelle TB1. Die Testblock-ID-Tabelle TB1 wird vom Prozessorkern **11** referenziert, wenn der Prozessorkern **11** einen RAM-Test auf dem RAM **21** ausführt. Wie in **Fig. 2** dargestellt, ist die Testblock-ID-Tabelle TB1 eine Tabelle, die eine ID-Nummer zur Identifizierung einer Vielzahl von Testblöcken und einen Adressbereich des Testblocks, der der ID-Nummer entspricht, speichert. In der in **Fig. 2** dargestellten Testblock-ID-Tabelle TB1 sind „1“, „2“, „3“, „4“ und „5“, die in der linken Spalte der Tabelle beschrieben sind, ID-Nummern.

[0023] Als nächstes wird der Ablauf eines anfänglichen Einstellprozesses beschrieben, der vom Prozessorkern **11** ausgeführt wird. Der Erst-Einstellprozess ist ein Prozess, der unmittelbar nach der Aktivierung des Prozessorkerns **11** gestartet wird. Wenn der Anfangseinstellprozess ausgeführt wird, speichert der Prozessorkern **11** „1“ im Blockbefehlswert TestBlock, der im RAM **21** in S10 bereitgestellt wird, und beendet den Anfangseinstellprozess, wie in **Fig. 3** dargestellt.

[0024] Als nächstes wird der Ablauf des vom Prozessorkern **11** ausgeführten RAM-Testprozesses beschrieben. Der RAM-Testprozess ist ein Prozess, der jedes Mal gestartet wird, wenn ein voreingestellter Ausführungszyklus verstrichen ist. In der vorliegenden Ausführungsform ist der Ausführungszyklus des RAM-Testprozesses z.B. auf 10 ms eingestellt.

[0025] Wenn der RAM-Testprozess ausgeführt wird, setzt der Prozessorkern **11** zunächst ein Leseverbot für die Prozessorkerne **12** und **13** in **S110**, wie in **Fig. 4** dargestellt. Konkret gibt der Prozessorkern **11** an die MPU **15** eine Verbotseinstellanforderung zur Verhinderung eines Lesezugriffs von den Prozessorkernen **12** und **13** auf die Aktualisierungsdaten **31** im RAM **21** aus. Wenn diese Verbotseinstellanforderung an die MPU **15** eingegeben wird, setzt die MPU **15** in einem in der MPU **15** vorgesehenen Speicherschutzeinstellregister ein Lesezugriffsverbot, um einen Lesezugriff auf die Aktualisierungsdaten **31** im RAM **21** von den Prozessorkernen **12** und **13** zu verbieten.

[0026] Als nächstes bezieht sich der Prozessorkern **11** in **S120** auf die im ROM **14** gespeicherte Testblock-ID-Tabelle TB1 und erfasst den Adressbereich des Testblocks, der der ID-Nummer entspricht, die mit dem im Blockbefehlswert TestBlock gespeicherten Wert übereinstimmt.

[0027] Dann führt der Prozessorkern **11** in **S130** einen RAM-Test auf dem RAM **21** für den in **S120** erworbenen Adressbereich aus. Im RAM-Test schreibt der Prozessorkern **11** die invertierten Daten, die durch Invertieren des Wertes der aus dem RAM **21** gelesenen Schreibdaten erhalten wurden, in den RAM **21**, liest die geschriebenen Daten und bestimmt, ob die invertierten Daten der gelesenen Daten mit den Schreibdaten übereinstimmen. Auf diese Weise wird der Ausfall des RAM **21** diagnostiziert. Nachdem die Diagnose durchgeführt wurde, schreibt der Prozessorkern **11** den ursprünglichen Wert in das RAM **21**.

[0028] Als nächstes speichert der Prozessorkern **11** in **S140** einen Mehrwert, der durch Addition von 1 zu dem im Blockbefehlswert TestBlock gespeicherten Wert als Blockbefehlswert TestBlock erhalten wird. Dann, in **S150**, bestimmt der Prozessorkern **11**, ob in den Handshake-Daten **41**, **51** „Leseanforderung vorhanden“ (auch Leseanforderungsanwesenheit genannt) eingestellt ist. Wenn „Leseanforderung nicht vorhanden“ (auch Leseanforderungsabwesenheit genannt) in den Handshake-Daten **41**, **51** eingestellt ist, bestimmt der Prozessorkern **11** in **S160**, ob der im Blockbefehlswert TestBlock gespeicherte Wert größer als die voreingestellte Tabellenblocknummer MaxBlock ist. In der vorliegenden Ausführungsform wird die Anzahl der Tabellenblöcke MaxBlock auf 5 gesetzt.

[0029] Wenn hier der im Blockbefehlswert TestBlock gespeicherte Wert gleich oder kleiner als die Anzahl der Tabellenblöcke MaxBlock ist, fährt der Prozessorkern **11** mit **S120** fort. Wenn andererseits der im Blockbefehlswert TestBlock gespeicherte Wert größer als die Anzahl der Tabellenblöcke MaxBlock ist, speichert der Prozessorkern **11** **1** im Blockbefehlswert TestBlock in S170 und fährt mit **S180** fort.

[0030] Wenn in **S150** bestimmt wird, dass in mindestens einem der Handshake-Daten **41** und **51** „Leseanforderung vorhanden (es liegt eine Leseanforderung vor)“ gesetzt ist, fährt der Prozessorkern **11**

mit **S180** fort. Wenn der Prozess dann zu **S180** fortschreitet, setzt der Prozessorkern **11** eine Leseberechtigung für die Prozessorkerne **12**, **13**. Konkret gibt der Prozessorkern **11** an die MPU **15** eine Berechtigungseinstellanforderung aus, um einen Lesezugriff von den Prozessorkernen **12** und **13** auf die Aktualisierungsdaten **31** im RAM **21** zu erlauben. Wenn diese Berechtigungseinstellanforderung an die MPU **15** eingegeben wird, setzt die MPU **15** eine Lesezugriffsberechtigung im Speicherschutzeinstellregister, das in der MPU **15** vorgesehen ist, um einen Lesezugriff von den Prozessorkernen **12** und **13** auf die Aktualisierungsdaten **31** im RAM **21** zuzulassen.

[0031] Weiterhin setzt der Prozessorkern **11** in **S190** „Leseanforderung nicht vorhanden“ in den Handshake-Daten **41**, **51** und beendet den RAM-Testprozess. Als nächstes wird eine Prozedur eines Zugriffsfehlerprozesses beschrieben, der vom Prozessorkern **12** ausgeführt wird. Der Zugriffsfehlerprozess ist ein Prozess, der gestartet wird, wenn eine Lesezugriffsverbotsmeldung von der MPU **15** eingegeben wird.

[0032] Wenn der Zugriffsfehlerprozess ausgeführt wird, setzt der Prozessorkern **12** zunächst „Leseanforderung vorhanden“ in den Handshake-Daten **41** in **S310**, wie in **Fig. 5** dargestellt. Dann bestimmt der Prozessorkern **12** in **S320**, ob „Leseanforderung vorhanden“ in den Handshake-Daten **41** gesetzt ist. Wenn „Leseanforderung vorhanden“ eingestellt ist, wartet der Prozessorkern **12**, bis „Leseanforderung nicht vorhanden“ in den Handshake-Daten **41** eingestellt ist, indem er die Verarbeitung in **S320** wiederholt.

[0033] Wenn dann in den Handshake-Daten **41** „Leseanforderung nicht vorhanden“ eingestellt ist, fährt der Prozessorkern **12** mit S330 fort. In **S330** kehrt der Prozess zu der Verarbeitung zurück, bei der der Fehler aufgetreten ist (d.h. die Verarbeitung, die die Benachrichtigung über das Eingabe-Lesezugriffsverbot verursacht hat), und der Zugriffsfehlerprozess wird beendet.

[0034] Der Zugriffsfehlerprozess, der vom Prozessorkern **13** ausgeführt wird, ist derselbe wie der Zugriffsfehlerprozess, der vom Prozessorkern **12** ausgeführt wird, mit der Ausnahme, dass die Handshake-Daten **41** in die Handshake-Daten **51** geändert werden und somit eine detaillierte Beschreibung ausgelassen wird.

[0035] Darüber hinaus führen die Prozessorkerne **12** und **13** auch einen anfänglichen Einstellprozess und einen RAM-Testprozess aus. Der Anfangseinstellprozess und der RAM-Testprozess des Prozessorkerns **12** unterscheiden sich vom Prozessorkern **11** dadurch, dass die Aktualisierungsdaten **32** und die Handshake-Daten **42** und **52** des RAM **22** anstelle der Aktualisierungsdaten **31** und der Handshake-Daten **41** und **51** des RAM **21** anvisiert werden.

[0036] In ähnlicher Weise unterscheiden sich der anfängliche Einstellprozess und der RAM-Testprozess des Prozessorkerns **13** vom Prozessorkern **11** dadurch, dass die Aktualisierungsdaten **33** und die Handshake-Daten **43** und **53** des RAM **23** anstelle der Aktualisierungsdaten **31** und der Handshake-Daten **41** und **51** des RAM **21** anvisiert werden.

[0037] Die Prozessorkerne **11** und **13** führen einen Zugriffsfehlerprozess auf den RAM **22** in der gleichen Weise aus wie die Prozessorkerne **12** und **13** einen Zugriffsfehlerprozess auf den RAM **21**. In ähnlicher Weise führen die Prozessorkerne **11** und **12** jeweils einen Zugriffsfehlerprozess für den RAM **23** aus.

[0038] Anschließend wird ein erstes konkretes Beispiel für eine Zugriffsentscheidung durch die MPU **15** beschrieben. Wie in **Fig. 6** gezeigt, setzt der Prozessorkern **11** zunächst ein Lesezugriffsverbot, um einen Lesezugriff auf die Aktualisierungsdaten **31** von den Prozessorkernen **12** und **13** zu verbieten. Der Pfeil **L1** zeigt an, dass der Prozessorkern **11** eine Aufforderung zum Setzen eines Verbots an die MPU **15** ausgibt.

[0039] Als nächstes aktualisiert der Prozessorkern **11** die Aktualisierungsdaten **31**, wie durch den Pfeil **L2** angezeigt. Dann setzt der Prozessorkern **11**, wie durch den Pfeil **L3** angezeigt, eine Lesezugriffsberechtigung, um einen Lesezugriff auf die Aktualisierungsdaten **31** von den Prozessorkernen **12** und **13** zu ermöglichen. Der Pfeil **L3** zeigt an, dass der Prozessorkern **11** eine Berechtigungseinstellanforderung an die MPU **15** ausgibt.

[0040] Danach gibt der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L4** zeigt an, dass der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** anfordert.

[0041] Da hier die Lesezugriffsberechtigung in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsberechtigungsmeldung an den Prozessorkern **12** aus. Beim Erwerb der Lesezugriffsberechtigungsmeldung von der MPU **15** greift der Prozessorkern **12** auf die Aktualisierungsdaten **31** zu und liest die Daten aus den Aktualisierungsdaten **31**. Der Pfeil **L5** zeigt an, dass der Prozessorkern **12** die Daten aus den Aktualisierungsdaten **31** liest.

[0042] Ferner gibt der Prozessorkern **13** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L6** zeigt an, dass der Prozessorkern **13** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** anfordert.

[0043] Da hier die Lesezugriffsberechtigung in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsberechtigungsmeldung an den Prozessorkern **13** aus. Beim Erwerb der Lesezugriffsberechtigungsmeldung von der MPU **15** greift der Prozessorkern **13** auf die Aktualisierungsdaten **31** zu und liest die Daten aus den Aktualisierungsdaten **31**. Der Pfeil **L7** zeigt an, dass der Prozessorkern **13** die Daten aus den Aktualisierungsdaten **31** liest.

[0044] Anschließend wird ein zweites konkretes Beispiel für eine Zugriffsentscheidung durch die MPU **15** beschrieben. Wie in **Fig. 7** gezeigt, setzt der Prozessorkern **11** zunächst ein Lesezugriffsverbot, um einen Lesezugriff auf die Aktualisierungsdaten **31** von den Prozessorkernen **12** und **13** zu verbieten. Der Pfeil **L11** zeigt an, dass der Prozessorkern **11** eine Aufforderung zum Setzen eines Verbots an die MPU **15** ausgibt.

[0045] Dann beginnt der Prozessorkern **11** mit der Ausführung des RAM-Testprozesses. Das Quadrat **SQ1** zeigt an, dass der Prozessorkern **11** den RAM-Testprozess ausführt. Nach dem Start des RAM-Testprozesses gibt der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L12** zeigt an, dass der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** anfordert.

[0046] Da hier das Lesezugriffsverbot in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsverbotsmeldung an den Prozessorkern **12** aus, was durch den Pfeil **L13** angezeigt wird. Wenn der Prozessorkern **12** die Mitteilung über das Lesezugriffsverbot von der MPU **15** erhält, beginnt der Prozessorkern **12** mit der Ausführung des Zugriffsfehlerprozesses und setzt zunächst „Leseanforderung vorhanden“ in den Handshake-Daten **41**, wie durch den Pfeil **L14** angezeigt.

[0047] Danach bestätigt der Prozessorkern **12** die Handshake-Daten **41**, wie durch das Quadrat **SQ2** und den Pfeil **L15** angezeigt, und wartet, bis in den Handshake-Daten **41** „Leseanforderung nicht vorhanden“ eingestellt ist.

[0048] Auf der anderen Seite führt der Prozessorkern **11** den RAM-Test an den Aktualisierungsdaten **31** aus, wie durch den Pfeil **L16** angezeigt wird. Ferner bestimmt der Prozessorkern **11**, wie durch den Pfeil **L17** angezeigt, ob „Leseanforderung vorhanden“ in den Handshake-Daten **41**, **51** eingestellt ist.

[0049] Da „Leseanforderung vorhanden“ in den Handshake-Daten **41** gesetzt wird, setzt der Prozessorkern **11** eine Lesezugriffsberechtigung, um einen Lesezugriff auf die Aktualisierungsdaten **31** von den Prozessorkernen **12** und **13** zu ermöglichen. Der Pfeil **L18** zeigt an, dass der Prozessorkern **11** eine Berechtigungseinstellanforderung an die MPU **15** ausgibt.

[0050] Dann setzt der Prozessorkern **11** „Leseanforderung nicht vorhanden“ in den Handshake-Daten **41**, wie durch den Pfeil **L19** angezeigt. Wenn „Leseanforderung nicht vorhanden“ in den Handshake-Daten **41** eingestellt ist, beendet der Prozessorkern **12** den Zugriffsfehlerprozess und gibt eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L20** zeigt an, dass der Prozessorkern **12** einen Lesezugriff auf die Aktualisierungsdaten **31** anfordert.

[0051] Da hier die Lesezugriffsberechtigung in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsberechtigungsmeldung an den Prozessorkern **12** aus. Beim Erwerb der Lesezugriffsberechtigungsmeldung von der MPU **15** greift der Prozessorkern **12** auf die Aktualisierungsdaten **31** zu und liest die Daten aus den Aktualisierungsdaten **31**. Der Pfeil **L21** zeigt an, dass der Prozessorkern **12** die Daten aus den Aktualisierungsdaten **31** liest.

[0052] Ferner gibt der Prozessorkern **13** eine Lesezugriffsanforderung für die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L22** zeigt an, dass der Prozessorkern **13** eine Lesezugriffsanforderung für die Aktualisierungsdaten **31** anfordert.

[0053] Da hier die Lesezugriffsberechtigung in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsberechtigungsmeldung an den Prozessorkern **13** aus. Beim Erwerb der Lesezugriffsberechtigungsmeldung von der MPU **15** greift der Prozessorkern **13** auf die Aktualisierungsdaten **31** zu und liest die Daten aus den Aktualisierungsdaten **31**. Der Pfeil **L23** zeigt an, dass der Prozessorkern **13** die Daten aus den Aktualisierungsdaten **31** liest.

[0054] Als nächstes wird eine ausschließliche Steuerung mit einer Semaphore beschrieben. Obwohl der Mikrocomputer **2** nicht über eine Semaphore verfügt, wird zur Vereinfachung der Beschreibung der Semaphore davon ausgegangen, dass die Prozessorkerne **11** und **12** die Erfassung einer Semaphore und die Freigabe einer Semaphore durchführen.

[0055] Wie in **Fig. 8** dargestellt, wird zunächst zum Zeitpunkt t0 das Semaphor freigegeben. Der Block **B1** gibt einen Zeitraum an, in dem die Semaphore freigegeben wird. Dann, zum Zeitpunkt t1, erwirbt der Prozessorkern **11** die Semaphore, und der Prozessorkern **11** führt den RAM-Test auf dem Testblock, dessen ID-Nummer **1** ist, vom Zeitpunkt t1 bis zum Zeitpunkt t2 durch. Der Block **B2** gibt einen Zeitraum an, während dessen die Semaphore vom Prozessorkern **11** erworben wird. Der Block **B3** zeigt einen Zeitraum an, während dessen der RAM-Test an dem Testblock mit der ID-Nummer **1** durchgeführt wird.

[0056] Der Prozessorkern **11** führt den RAM-Test für den Testblock mit der ID-Nummer 2 vom Zeitpunkt t2 bis zum Zeitpunkt t3 durch. Der Block **B4** gibt einen Zeitraum an, während dessen der RAM-Test an dem Testblock mit der ID-Nummer 2 durchgeführt wird.

[0057] Der Prozessorkern **11** führt den RAM-Test für den Testblock mit der ID-Nummer 3 vom Zeitpunkt t3 bis zum Zeitpunkt t4 durch. Der Block **B5** gibt einen Zeitraum an, während dessen der RAM-Test für den Testblock mit der ID-Nummer 3 durchgeführt wird.

[0058] Der Prozessorkern **11** führt den RAM-Test für den Testblock mit der ID-Nummer 4 vom Zeitpunkt t4 bis zum Zeitpunkt t5 durch. Der Block **B6** gibt einen Zeitraum an, während dessen der RAM-Test an dem Testblock mit der ID-Nummer 4 durchgeführt wird.

[0059] Der Prozessorkern **11** führt den RAM-Test für den Testblock mit der ID-Nummer 5 vom Zeitpunkt t5 bis zum Zeitpunkt t6 durch. Der Block **B7** gibt einen Zeitraum an, während dessen der RAM-Test für den Testblock mit der ID-Nummer 5 durchgeführt wird.

[0060] Wenn der RAM-Test für den Testblock, dessen ID-Nummer 5 ist, zur Zeit t6 abgeschlossen ist, gibt der Prozessorkern **11** die Semaphore frei. Der Prozessorkern **12** versucht, die Semaphore vom Zeitpunkt t1 bis zum Zeitpunkt t6 zu erfassen, und schlägt fehl. Der Block **B8** gibt einen Zeitraum an, während dessen der Prozessorkern **12** wartet, bis er ein Semaphor erfasst hat.

[0061] Dann, zum Zeitpunkt t7, erwirbt der Prozessorkern **12** die Semaphore, und der Prozessorkern **12** liest die Daten aus den Aktualisierungsdaten **31** vom Zeitpunkt t7 bis zum Zeitpunkt t8. Der Block **B9** zeigt einen Zeitraum an, in dem die Semaphore freigegeben wird. Der Block **B10** zeigt einen Zeitraum an, in dem die Semaphore vom Prozessorkern **12** erfasst wird. Der Block **B11** zeigt einen Zeitraum an, während dessen der Prozessorkern **12** die Daten liest.

[0062] Wenn das Lesen der Daten zum Zeitpunkt t8 endet, gibt der Prozessorkern **12** die Semaphore frei. Der Block **B12** gibt einen Zeitraum an, während dessen die Semaphore freigegeben wird. Als nächstes wird eine ausschließliche Steuerung der vorliegenden Ausführungsform durch den Mikrocomputer **2** beschrieben.

[0063] Wie in **Fig. 9** dargestellt, setzt die MPU **15** zum Zeitpunkt t20 zunächst eine Leseberechtigung. Der Block **B21** zeigt einen Zeitraum an, in dem eine Lesezugriffserlaubnis gesetzt wird. Dann, zum Zeitpunkt t21, gibt der Prozessorkern **11** eine Aufforderung zum Setzen eines Verbots an die MPU **15** aus. Daraufhin setzt die MPU **15** ein Lesezugriffsverbot. Dann führt der Prozessorkern **11** den RAM-Test für den Testblock mit der ID-Nummer **1** bis zum Zeitpunkt t22 aus.

[0064] Wenn der RAM-Test für den Testblock, dessen ID-Nummer **1** ist, endet, gibt der Prozessorkern **11** eine Erlaubniseinstellanforderung an die MPU **15** aus. Dadurch setzt die MPU **15** eine Lesezugriffsberechtigung. Der Block **B22** gibt einen Zeitraum an, während dessen der RAM-Test für den Testblock mit der ID-Nummer **1** durchgeführt wird. Der Block **B23** gibt einen Zeitraum an, während dessen das Lesezugriffsverbot gesetzt wird.

[0065] Der Prozessorkern **11** gibt vom Zeitpunkt t21 bis zum Zeitpunkt t22 eine Lesezugriffsanforderung an die MPU **15** aus. Da das Lesezugriffsverbot jedoch in der MPU **15** festgelegt ist, wartet der Prozessorkern **12**, bis der Lesezugriff erlaubt wird. Dann, zum Zeitpunkt t23, gibt der Prozessorkern **12** eine Lesezugriffsanforderung an die MPU **15** aus. Zum Zeitpunkt t23 wird die Lesezugriffserlaubnis in der MPU **15** gesetzt, so dass der Prozessorkern **12** die Daten aus den Aktualisierungsdaten **31** liest. Der Block **B24** gibt einen Zeitraum an, in dem das Gerät wartet, bis der Lesezugriff gestattet wird. Der Block **B25** gibt eine Zeitspanne an, während der die Lesezugriffserlaubnis gesetzt ist. Der Block **B26** gibt einen Zeitraum an, während dessen der Prozessorkern **12** Daten liest.

[0066] Dann startet der Prozessorkern **11** zum Zeitpunkt t24 den RAM-Test des Testblocks mit der ID-Nummer **2** und zum Zeitpunkt t25 den RAM-Test des Testblocks mit der ID-Nummer **3**. Der Block **B27** gibt einen Zeitraum an, in dem der RAM-Test des Testblocks mit der ID-Nummer **2** durchgeführt wird.

[0067] Die wie oben beschrieben konfigurierte ECU **1** umfasst die Prozessorkerne **11**, **12** und **13**, das RAM **21** und die MPU **15**. Das RAM **21** speichert die Aktualisierungsdaten **31**, auf die jeder der Prozessorkerne **11**, **12** und **13** lesbar zugreifen kann.

[0068] Die MPU **15** steuert die Prozessorkerne **11**, **12** und **13** so, dass die Prozessorkerne **12** und **13** nicht auf die Aktualisierungsdaten **31** zugreifen können, wenn der Prozessorkern **11** auf die Aktualisierungsdaten **31** zugreift.

[0069] Wie oben beschrieben, steuert die MPU **15** in der ECU **1** die Prozessorkerne **11**, **12**, **13** so, dass die Prozessorkerne **12**, **13** nicht auf die Aktualisierungsdaten **31** zugreifen können, wenn der Prozessorkern **11** auf die Aktualisierungsdaten **31** zugreift. Das heißt, da die ECU **1** die MPU **15** anstelle einer Semaphore verwendet, ist eine spezielle Schnittstelle auf der Seite des Prozessorkerns, die auf die Daten zugreift, nicht erforderlich. Infolgedessen kann die ECU **1** eine Erhöhung des Overheads unterdrücken und eine Erhöhung des Programmcodes auf-

grund der exklusiven Steuerung zwischen den Prozessorkernen unterdrücken.

[0070] Die ECU **1** braucht keinen dedizierten gemeinsamen Speicher für den Datentransfer zwischen dem Prozessorkern **11** und den Prozessorkernen **12** und **13**. Jeder der Prozessorkerne **11**, **12** und **13** kann direkt auf den RAM **21** zugreifen, um Daten zwischen dem Prozessorkern **11** und den Prozessorkernen **12** und **13** zu übertragen.

[0071] Der Prozessorkern **11** setzt die MPU **15** auf ein Lesezugriffsverbot, um einen Lesezugriff von den Prozessorkernen **12** und **13** auf die Aktualisierungsdaten **31** während eines Zeitraums zu verhindern, in dem der Prozessorkern **11** auf die Aktualisierungsdaten **31** zugreift. Ferner setzt der Prozessorkern **11** die MPU **15** auf eine Lesezugriffsberechtigung, um einen Lesezugriff auf die Aktualisierungsdaten **31** von den Prozessorkernen **12** und **13** während eines Zeitraums, in dem der Prozessorkern **11** nicht auf die Aktualisierungsdaten **31** zugreift, zu erlauben.

[0072] Wenn eine Datenaktualisierung von einem spezifischen Prozessorkern durchgeführt wird, ist typischerweise das Schreiben von Daten von einem anderen Prozessorkern verboten, aber das Lesen von Daten ist nicht verboten. Der Grund dafür ist, dass das Lesen der Daten das Schreiben der Daten nicht beeinträchtigt.

[0073] Im Gegensatz dazu verwendet der ECU **1** die MPU **15**, um eine beabsichtigte Lesezugriffsanforderung zu erkennen, indem er absichtlich das Lesen von Daten aus einem anderen Prozessorkern verbietet. Normalerweise wird eine MPU verwendet, um einen unbeabsichtigten unberechtigten Zugriff zu erkennen. Im Gegensatz dazu kann die ECU **1** durch die Verwendung der MPU **15** zur Erkennung des beabsichtigten Zugriffs den Overhead reduzieren, der bei jeder Verwendung der Semaphore anfällt.

[0074] Der RAM **21** der ECU **1** speichert die Handshake-Daten **41**, **51**, die für das Handshake zwischen den Prozessorkernen **11**, **12**, **13** verwendet werden und auf die die Prozessorkerne **11**, **12**, **13** lesbar und aktualisierbar zugreifen können.

[0075] Wenn die Prozessorkerne **12** und **13** während eines Zeitraums, in dem der Prozessorkern **11** auf die Aktualisierungsdaten **31** zugreift, eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** ausgeben, gibt die MPU **15** eine Lesezugriffsverbotsmeldung an die Prozessorkerne **12** und **13** aus.

[0076] Jeder der Prozessorkerne **12** und **13** erhält von der MPU **15** eine Leseverbotsmeldung. Dann setzt jeder der Prozessorkerne **12** und **13** „Leseanforderung vorhanden“, was anzeigt, dass eine Lesezugriffsanforderung in den Handshake-Daten **41**, **51** vorhanden ist, die für das Handshake zwischen dem Prozessorkern **11** und den Prozessorkernen **12**, **13** verwendet werden.

[0077] Die Prozessorkerne **12** und **13** geben jeweils eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus, wenn die Handshake-Daten **41**, **51** von einem Zustand, in dem „Leseanforderung vorhanden“ gesetzt ist, in einen Zustand wechseln, in dem „Leseanforderung nicht vorhanden“ gesetzt ist.

[0078] Während des Zeitraums, in dem der Prozessorkern **11** auf die Aktualisierungsdaten **31** zugreift, bestätigt der Prozessorkern **11** die Handshake-Daten **41** und **51** jedes Mal wiederholt, wenn der RAM-Test für einen Testblock beendet wird. Dadurch wird bestimmt, ob in den Handshake-Daten **41**, **51** „Leseanforderung vorhanden“ gesetzt ist.

[0079] Wenn der Prozessorkern **11** bestimmt, dass in den Handshake-Daten **41** und **51** „Leseanforderung vorhanden“ gesetzt ist, unterbricht der Prozessorkern **11** den RAM-Test und setzt die MPU **15** auf die Lesezugriffsberechtigung.

[0080] Wenn der Prozessorkern **11** bestimmt, dass „Leseanforderung vorhanden“ in den Handshake-Daten **41**, **51** gesetzt ist, setzt der Prozessorkern **11** „Leseanforderung nicht vorhanden“, was anzeigt, dass keine Lesezugriffsanforderung in den Handshake-Daten **41** und **51** vorhanden ist.

[0081] Infolgedessen unterbricht das Steuergerät **1** den RAM-Test, wenn der Prozessorkern **11** eine Lesezugriffsanforderung von den Prozessorkernen **12** und **13** erkennt. Die Prozessorkerne **12** und **13** können so veranlasst werden, den Lesezugriff auf die Aktualisierungsdaten **31** durchzuführen. Daher kann das Steuergerät **1** die Zeit verkürzen, die die Prozessorkerne **12** und **13** auf einen Lesezugriff auf die Aktualisierungsdaten **31** warten. Die ECU **1** kann die Verarbeitungseffizienz der Prozessorkerne **12** und **13** verbessern.

[0082] In der oben beschriebenen Ausführungsform entspricht die ECU **1** einer elektronischen Steuereinheit, der RAM **21** einer gemeinsam genutzten Datenspeichereinheit und die MPU **15** einer Speicherschutzeinheit. Die Aktualisierungsdaten **31** entsprechen einer gemeinsam genutzten Dateneinheit. Der Prozessorkern **11** entspricht einem spezifischen Prozessorkern, und die Prozessorkerne **12** und **13** entsprechen verschiedenen Prozessorkernen.

[0083] **S110** entspricht der Verarbeitung als Verbotseinstelleinheit; **S180** entspricht der Verarbeitung als Erlaubniseinstelleinheit. Ein Lesezugriffsverbot ent-

spricht einem Zugriffsverbot; eine Lesezugriffserlaubnis entspricht einer Zugriffserlaubnis.

[0084] Das RAM **21** entspricht einer Handshake-Speichereinheit. Die Ausgabe der Lesezugriffsverbotsmeldung entspricht einem Speicherzugriffsverletzungsprozess. **S310** entspricht der Verarbeitung als Anforderungsanwesenheitseinstelleinheit; **S320** und **S330** entsprechen der Verarbeitung als Wiederzugriffseinheit. Die Ausgabe der „Leseanforderung vorhanden“ entspricht einer Zugriffsanforderungsanwesenheit.

[0085] Darüber hinaus entspricht **S150** der Verarbeitung als Anforderungsbestimmungseinheit und Unterbrechungseinheit; **S190** entspricht der Verarbeitung als Anforderungsabwesenheitseinstelleinheit. „Beenden des RAM-Tests für einen Testblock“ entspricht einer Bestätigungsbedingung. Der RAM-Test entspricht einem Zugriffsprozess; „Leseanforderung nicht vorhanden“ entspricht einer Abwesenheit der Zugriffsanforderung.

[Zweite Ausführungsform]

[0086] Im Folgenden wird eine zweite Ausführungsform der vorliegenden Offenbarung unter Bezugnahme auf die Zeichnungen beschrieben. Beachten Sie, dass in der zweiten Ausführungsform Teile beschrieben werden, die sich von der ersten Ausführungsform unterscheiden. Für gemeinsame Komponenten werden die gleichen Bezugszeichen angegeben.

[0087] Die ECU **1** der zweiten Ausführungsform unterscheidet sich von der ersten Ausführungsform dadurch, dass ein erster Prozessorprozess und ein Datenleseprozess sowie eine Prioritätstabelle **TB2** hinzugefügt werden. Im ROM **14** ist eine Prioritätstabelle **TB2** gespeichert. Wie in **Fig. 10** dargestellt, ist die Prioritätstabelle **TB2** eine Tabelle, die eine Prozessorkernnummer zur Identifizierung der Prozessorkerne **11**, **12** und **13** sowie eine der Prozessorkernnummer entsprechende Priorität speichert. Die Prozessorkernnummern „**1**“, „**2**“ und „**3**“ entsprechen den Prozessorkernen **11**, **12** bzw. **13**. In der in **Fig. 10** dargestellten Prioritätstabelle **TB2** ist die Priorität des Prozessorkerns **11** auf „Hoch“, die Priorität des Prozessorkerns **12** auf „Mittel“ und die Priorität des Prozessorkerns **13** auf „Niedrig“ gesetzt.

[0088] Als nächstes wird die Prozedur des vom Prozessorkern **11** ausgeführten Erstprozessorprozesses beschrieben. Der Erst-Prozessor-Prozess ist ein Prozess, der jedes Mal gestartet wird, wenn ein voreingestellter Ausführungszyklus abläuft. In der vorliegenden Ausführungsform wird der Ausführungszyklus eines Erst-Prozessor-Prozesses z.B. auf 10 ms eingestellt. Ferner ist der Erst-Prozessor-Prozess ein Prozess zur sequentiellen Ausführung von N geteilten Prozessen (im Folgenden als Teilungsprozess bezeichnet). N ist eine ganze Zahl von 2 oder mehr.

[0089] Wenn der erste Prozessorprozess ausgeführt wird, setzt der Prozessorkern **11**, wie in **Fig. 11** dargestellt, zunächst ein Leseverbot für die Prozessorkerne **12**, **13** in **S410** auf die gleiche Weise wie in **S110**.

[0090] In **S420** speichert der Prozessorkern **11 1** im Verarbeitungsbefehlswert i, der im RAM **21** bereitgestellt wird. Dann führt der Prozessorkern **11** in **S430** den i-ten Teilungsprozess aus. Ferner speichert der Prozessorkern **11** in **S440** einen Mehrwert, der durch Addition von 1 zu dem in dem Verarbeitungsbefehlswert i gespeicherten Wert als Verarbeitungsbefehlswert i erhalten wird.

[0091] Als nächstes überprüft der Prozessorkern **11** in **S450** die Handshake-Daten **41**, **51**. Dann, in **S460**, bestimmt der Prozessorkern **11**, ob in mindestens einem der Handshake-Daten **41** und **51** „Leseanforderung vorhanden“ gesetzt ist. Wenn hier „Leseanforderung abwesend“ in den Handshake-Daten **41**, **51** eingestellt ist, bestimmt der Prozessorkern **11** in **S470**, ob der in der Verarbeitungsanweisung gespeicherte Wert i größer als die voreingestellte Anzahl N von Divisionsprozessen ist.

[0092] Wenn hier der in der Verarbeitungsanweisung gespeicherte Wert i gleich oder kleiner als die Anzahl N der Divisionsprozesse ist, geht der Prozessorkern **11** zu **S430** über. Wenn andererseits der im Verarbeitungsbefehlswert i gespeicherte Wert größer ist als die Anzahl der Divisionsprozesse, setzt der Prozessorkern **11** in **S480** auf die gleiche Weise wie in **S180** eine Leseberechtigung für die Prozessorkerne **12** und **13**. Der erste Prozessorprozess wird beendet.

[0093] Wenn „Leseanforderung vorhanden“ in mindestens einem der Handshake-Daten **41** und **51** in **S460** eingestellt ist, bezieht sich der Prozessorkern **11** auf die Prioritätstabelle **TB2** in **S490**.

[0094] Dann, in **S500**, setzt der Prozessorkern **11** eine Leseberechtigung für den Prozessorkern mit der höchsten Priorität unter den Prozessorkernen, die den Handshake-Daten entsprechen, für die „Leseanforderung vorhanden“ gesetzt ist. Wenn zum Beispiel die Prozessorkerne, die den Handshake-Daten entsprechen, für die „Leseanforderung vorhanden“ eingestellt ist, die Prozessorkerne **12** und **13** sind, setzt der Prozessorkern **11** die Leseberechtigung für den Prozessorkern **12**.

[0095] Darüber hinaus setzt der Prozessorkern **11** in **S510** die „Leseanforderung nicht vorhanden“ auf die Handshake-Daten für den Prozessorkern, der unter den Prozessorkernen, die den Handshake-Daten entsprechen, für die „Leseanforderung vorhanden“

eingestellt ist, die höchste Priorität hat. Dann endet der erste Prozessorprozess.

[0096] Als nächstes wird der Ablauf eines Datenleseprozesses beschrieben, der vom Prozessorkern **12** ausgeführt wird. Der Datenleseprozess ist ein Prozess, der gestartet wird, wenn im Prozessorkern **12** eine Datenleseanforderung an die Aktualisierungsdaten **31** erfolgt.

[0097] Wenn der Datenlesevorgang ausgeführt wird, führt der Prozessorkern **12** zunächst einen Datenlesezugriff in **S610** durch, wie in **Fig. 12** dargestellt. Konkret gibt der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus.

[0098] Dann, in **S620**, bestimmt der Prozessorkern **12**, ob ein Zugriffsfehler vorliegt. Konkret stellt der Prozessorkern **12** fest, dass ein Zugriffsfehler vorliegt, wenn er die Mitteilung über das Lesezugriffsverbot von der MPU **15** erhält, und er stellt fest, dass kein Zugriffsfehler vorliegt, wenn er die Mitteilung über die Lesezugriffserlaubnis von der MPU **15** erhält.

[0099] Hier setzt der Prozessorkern **12** bei einem Zugriffsfehler „Leseanforderung vorhanden“ in den Handshake-Daten **41** in **S630**. Dann, in **S640**, bestimmt der Prozessorkern **12**, ob „Leseanforderung vorhanden“ in den Handshake-Daten **41** gesetzt ist. Wenn hier „Leseanforderung vorhanden“ eingestellt ist, wartet der Prozessorkern **12**, bis „Leseanforderung nicht vorhanden“ in den Handshake-Daten **41** eingestellt ist, indem er die Verarbeitung in **S640** wiederholt.

[0100] Wenn dann in den Handshake-Daten **41** „Leseanforderung vorhanden“ eingestellt ist, geht der Prozess zu **S650** über. Wenn kein Zugriffsfehler in **S620** vorliegt, fährt der Prozess mit **S650** fort.

[0101] Wenn dann zu **S650** übergegangen wird, greift der Prozessorkern **12** auf die Aktualisierungsdaten **31** zu und liest die Daten aus den Aktualisierungsdaten **31**. Dann prüft der Prozessorkern **12** in **S660** die Handshake-Daten **51**.

[0102] Dann bestimmt in **S670** der Prozessorkern **12**, ob in den Handshake-Daten **51** „Leseanforderung vorhanden“ eingestellt ist. Wenn „Leseanforderung nicht vorhanden“ in den Handshake-Daten **51** gesetzt ist, setzt der Prozessorkern **12** die Leseberechtigung für den Prozessorkern **13** in S680 und beendet den Datenlesevorgang.

[0103] Wenn „Leseanforderung vorhanden“ in den Handshake-Daten **51** in **S670** eingestellt ist, bezieht sich der Prozessorkern **12** auf die Prioritätstabelle **TB2** in S690. Dann setzt der Prozessorkern **12** in **S700** eine Leseberechtigung für den Prozessorkern mit der höchsten Priorität unter den Prozessorkernen, die den Handshake-Daten entsprechen, für die „Leseanforderung vorhanden“ gesetzt ist. Wenn z.B. der Prozessorkern, der den Handshake-Daten entspricht, für die „Leseanforderung vorhanden“ gesetzt ist, der Prozessorkern **13** ist, setzt der Prozessorkern **12** die Leseberechtigung für den Prozessorkern **13**.

[0104] Darüber hinaus setzt der Prozessorkern **12** in **S710** die „Leseanforderung nicht vorhanden“ auf die Handshake-Daten für den Prozessorkern mit der höchsten Priorität unter den Prozessorkernen, die den Handshake-Daten entsprechen, für die die „Leseanforderung vorhanden“ eingestellt ist. Dann endet der Datenleseprozess.

[0105] Der vom Prozessorkern **13** ausgeführte Datenleseprozess ist derselbe wie der vom Prozessorkern **12** ausgeführte Datenleseprozess mit dem Unterschied, dass die Handshake-Daten **41** in die Handshake-Daten **51** geändert werden und somit die detaillierte Beschreibung wegfällt.

[0106] Sodann wird ein konkretes Beispiel für eine Zugriffsentscheidung durch die MPU **15** nach der zweiten Ausführungsform beschrieben. Wie in **Fig. 13** dargestellt, setzt der Prozessorkern **11** zunächst ein Lesezugriffsverbot, um einen Lesezugriff auf die Aktualisierungsdaten **31** von den Prozessorkernen **12** und **13** zu verbieten. Der Pfeil **L31** zeigt an, dass der Prozessorkern **11** eine Aufforderung zum Setzen eines Verbots an die MPU **15** ausgibt.

[0107] Dann beginnt der Prozessorkern **11** mit der sequentiellen Ausführung der N Teilungsprozesse. Das Quadrat **SQ3** zeigt an, dass der Prozessorkern **11** sequentiell N Teilungsprozesse ausführt. Nachdem die sequentielle Ausführung der N Teilungsprozesse gestartet wurde, gibt der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L32** zeigt an, dass der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** anfordert.

[0108] Da hier das Lesezugriffsverbot in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsverbotsmeldung an den Prozessorkern **12** aus, was durch den Pfeil **L33** angezeigt wird. Nach Erhalt der Lesezugriffsverbotsmeldung von der MPU **15** setzt der Prozessorkern **12** „Leseanforderung vorhanden“ auf die Handshake-Daten **41**, wie durch den Pfeil **L34** angezeigt.

[0109] Danach bestätigt der Prozessorkern **12** die Handshake-Daten **41**, wie durch das Quadrat **SQ4** und den Pfeil **L35** angezeigt, und wartet, bis in den Handshake-Daten **41** „Leseanforderung nicht vorhanden“ eingestellt ist.

[0110] Ferner gibt der Prozessorkern **13** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L36** zeigt an, dass der Prozessorkern **13** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** anfordert.

[0111] Da hier das Lesezugriffsverbot in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsverbotsmeldung an den Prozessorkern **13** aus, wie durch den Pfeil **L37** angezeigt wird. Nach Erhalt der Lesezugriffsverbotsmeldung von der MPU **15** setzt der Prozessorkern **13** „Leseanforderung vorhanden“ auf die Handshake-Daten **51**, wie durch den Pfeil **L38** angezeigt.

[0112] Danach bestätigt der Prozessorkern **13** die Handshake-Daten **51**, wie durch das Quadrat **SQ5** und den Pfeil **L39** angezeigt, und wartet, bis in den Handshake-Daten **51** „Leseanforderung nicht vorhanden“ eingestellt ist.

[0113] Auf der anderen Seite führt der Prozessorkern **11** sequentiell die N-Abteilungsprozesse aus und aktualisiert die Aktualisierungsdaten **31**, wie durch den Pfeil **L40** angezeigt. Ferner bestimmt der Prozessorkern **11**, wie durch den Pfeil **L41** angezeigt, ob in den Handshake-Daten **41** und **51** „Leseanforderung vorhanden“ gesetzt ist.

[0114] Da hier in den Handshake-Daten **41** und **51** „Leseanforderung vorhanden“ eingestellt ist, bezieht sich der Prozessorkern **11** auf die im ROM **14** gespeicherte Prioritätstabelle **TB2**, wie durch den Pfeil **L42** angezeigt.

[0115] Dann legt der Prozessorkern **11** die Lesezugriffsberechtigung fest, um den Lesezugriff von dem Prozessorkern **12** zu erlauben, der unter den Prozessorkernen **12** und **13** die höchste Priorität hat. Der Pfeil **L43** zeigt an, dass der Prozessorkern **11** eine Berechtigungseinstellanforderung des Prozessorkerns **12** an die MPU **15** ausgibt.

[0116] Dann setzt der Prozessorkern **11** „Leseanforderung nicht vorhanden“ in den Handshake-Daten **41**, wie durch den Pfeil **L44** angezeigt. Wenn „Leseanforderung nicht vorhanden“ in den Handshake-Daten **41** eingestellt ist, gibt der Prozessorkern **12** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L45** zeigt an, dass der Prozessorkern **12** einen Lesezugriff auf die Aktualisierungsdaten **31** anfordert.

[0117] Da hier die Lesezugriffsberechtigung in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsberechtigungsmeldung an den Prozessorkern **12** aus. Beim Erwerb der Lesezugriffsberechtigungsmeldung von der MPU **15** greift der Prozessorkern **12** auf die Aktualisierungsdaten **31** zu und liest die Daten aus den Aktualisierungsdaten **31**. Der Pfeil **L46** zeigt an, dass der Prozessorkern **12** die Daten aus den Aktualisierungsdaten **31** liest.

[0118] Da hier „Leseanforderung vorhanden“ in den Handshake-Daten **51** eingestellt ist, bezieht sich der Prozessorkern **12** auf die im ROM **14** gespeicherte Prioritätstabelle **TB2**, wie durch den Pfeil **L47** angezeigt.

[0119] Dann legt der Prozessorkern **12** eine Lesezugriffsberechtigung fest, um einen Lesezugriff vom Prozessorkern **13** zu ermöglichen. Der Pfeil **L48** zeigt an, dass der Prozessorkern **12** eine Berechtigungseinstellanforderung des Prozessorkerns **13** an die MPU **15** ausgibt.

[0120] Dann setzt der Prozessorkern **12** „Leseanforderung nicht vorhanden“ auf die Handshake-Daten **51**, wie durch den Pfeil **L49** angezeigt. Wenn „Leseanforderung nicht vorhanden“ in den Handshake-Daten **51** eingestellt ist, gibt der Prozessorkern **13** eine Lesezugriffsanforderung auf die Aktualisierungsdaten **31** an die MPU **15** aus. Der Pfeil **L50** zeigt an, dass der Prozessorkern **13** einen Lesezugriff auf die Aktualisierungsdaten **31** anfordert.

[0121] Da hier die Lesezugriffsberechtigung in der MPU **15** eingestellt ist, gibt die MPU **15** eine Lesezugriffsberechtigungsmeldung an den Prozessorkern **13** aus. Beim Erwerb der Lesezugriffsberechtigungsmeldung von der MPU **15** greift der Prozessorkern **13** auf die Aktualisierungsdaten **31** zu und liest die Daten aus den Aktualisierungsdaten **31**. Der Pfeil **L51** zeigt an, dass der Prozessorkern **13** die Daten aus den Aktualisierungsdaten **31** liest.

[0122] Der wie oben beschrieben konfigurierte ECU **1** enthält eine Prioritätstabelle **TB2**, in der die Priorität für jeden der Prozessorkerne **11**, **12** und **13** festgelegt ist. Dann setzt der Prozessorkern **11** auf der Grundlage der Prioritätstabelle **TB2** die MPU **15** auf Lesezugriffsberechtigung für den Prozessorkern **12** mit der höchsten Priorität unter den Prozessorkernen **12** und **13**, die die Lesezugriffsanforderung an die MPU **15** ausgeben.

[0123] Ferner setzt der Prozessorkern **11** in den Handshake-Daten **41**, die für den Handshake zwischen dem Prozessorkern **11** und dem Prozessorkern **12** verwendet werden, „Leseanforderung nicht vorhanden“. Auf diese Weise kann die ECU **1** dem Prozessorkern **12** mit einer höheren Priorität den Zugriff auf die Aktualisierungsdaten **31** anstelle des Prozessorkerns **13** mit einer niedrigeren Priorität ermöglichen. Auf diese Weise kann die ECU **1** das Auftreten einer Situation unterdrücken, in der die Ausführung des Prozesses mit der höheren Priorität aufgrund der Ausführung des Prozesses mit der niedrigeren Priorität verzögert wird, und die Reaktionsfähigkeit der ECU **1** kann verbessert werden.

[0124] In der oben beschriebenen Ausführungsform entspricht **S410** der Verarbeitung als Verbotseinstelleinheit; **S480** entspricht der Verarbeitung als Erlaubniseinstelleinheit. Außerdem entspricht **S630** der Verarbeitung als Einstelleinheit; **S640** entspricht der Verarbeitung als Einheit für den erneuten Zugriff.

[0125] Darüber hinaus entspricht **S460** der Verarbeitung als Anforderungsbestimmungseinheit, **S460**, **S490** und **S500** entsprechen der Verarbeitung als Unterbrechungseinheit; **S510** entspricht der Verarbeitung als Anforderungsabwesenheitseinstelleinheit. „Beenden eines Teilungsprozesses“ entspricht einer Bestätigungsbedingung; der Teilungsprozess entspricht einem Zugriffsprozess.

[Dritte Ausführungsform]

[0126] Im Folgenden wird eine dritte Ausführungsform der vorliegenden Offenbarung unter Bezugnahme auf die Zeichnungen beschrieben. In der dritten Ausführungsform werden Teile beschrieben, die sich von der ersten Ausführungsform unterscheiden. Gemeinsame Komponenten werden mit den gleichen Bezugszeichen versehen.

[0127] Wie in **Fig. 14** dargestellt, unterscheidet sich die ECU **1** gemäß der dritten Ausführungsform von der ersten Ausführungsform in (i) dem Punkt, dass ein Peripheriegerät **17** hinzugefügt wird, (ii) dem Punkt, dass die RAMs **21**, **22**, **23** weggelassen werden, während ein RAM **26** hinzugefügt wird, und (iii) dem Punkt, dass das ROM **14** ein Peripheriegerät-Steuerprogramm **71** zur Steuerung eines Peripheriegeräts **17** speichert.

[0128] Das Peripheriegerät **17** und das RAM **26** sind an den Systembus **16** angeschlossen. Das Peripheriegerät **17** gibt ein PWM-Signal an das Stellglied **4** entsprechend einer Anweisung von den Prozessorkernen **11**, **12** und **13** aus. PWM ist eine Abkürzung für Pulsweitenmodulation. In der vorliegenden Ausführungsform ist das Stellglied **4** eine elektronische Drosselklappe.

[0129] Der RAM **26** speichert die Handshake-Daten **61**, **62**, **63**. Die Handshake-Daten **61** sind Daten, in denen die Anwesenheit (auch Vorhandensein genannt) oder die Abwesenheit (auch Nichtvorhandensein genannt) der Ausführungsanforderung des Peripheriegerät-Steuerprogramms **71** durch den Prozessorkern **11** festgelegt wird. Bei den Handshake-Daten **62** handelt es sich um Daten, in denen das Vorhandensein oder Nichtvorhandensein der Ausführungsanforderung des Peripheriegerät-Steuerprogramms **71** durch den Prozessorkern **12** festgelegt wird. Die Handshake-Daten **63** sind Daten, in denen das Vorhandensein oder Nichtvorhandensein der Ausführungsanforderung des Peripheriegerät-Steuerprogramms **71** durch den Prozessorkern **13** festgelegt wird.

[0130] Darüber hinaus unterscheiden sich bei der ECU **1** der dritten Ausführungsform die folgenden ersten und zweiten Unterschiede von der ersten Ausführungsform beim Datenzugriff. Der erste Unterschied besteht darin, dass die Prozessorkerne **11**, **12** und **13** auf das Peripheriegerätesteuerprogramm **71** im ROM **14** zugreifen, anstatt auf die Aktualisierungsdaten **31** im RAM **21** zuzugreifen. Der zweite Unterschied besteht darin, dass die Prozessorkerne **12** und **13** auf die Handshake-Daten **62** und **63** im RAM **26** zugreifen, anstatt auf die Handshake-Daten **41** und **51** im RAM **21** zuzugreifen.

[0131] Wie oben beschrieben, ähnelt die ECU **1** nach der dritten Ausführungsform der ersten Ausführungsform in Bezug auf den Datenzugriff, weshalb auf die Beschreibung eines Verarbeitungsverfahrens anhand eines Flussdiagramms verzichtet wird. Als nächstes wird ein erstes konkretes Beispiel für eine Execution Access Arbitration durch die MPU **15** beschrieben.

[0132] Wie in **Fig. 15** gezeigt, legt der Prozessorkern **11** zunächst ein Ausführungszugriffsverbot fest, um einen Ausführungszugriff von den Prozessorkernen **12** und **13** auf das Peripheriegerätesteuerungsprogramm **71** zu verbieten. Der Pfeil **L61** zeigt an, dass der Prozessorkern **11** eine Verbotseinstellanforderung an die MPU **15** ausgibt.

[0133] Als nächstes holt der Prozessorkern **11** das Peripheriegerät-Steuerprogramm **71** aus dem ROM **14**, wie durch den Pfeil **L62** angezeigt. Dann führt der Prozessorkern **11** die Steuerung zur Ausgabe eines PWM-Signals an das Stellglied **4** aus, wie durch den Pfeil **L63** angezeigt. Ferner legt der Prozessorkern **11**, wie durch den Pfeil **L64** angezeigt, eine Ausführungszugriffsberechtigung fest, um einen Ausführungszugriff von den Prozessorkernen **12** und **13** auf das Peripheriegerät-Steuerprogramm **71** zu ermöglichen. Der Pfeil **L64** zeigt an, dass der Prozessorkern **11** eine Berechtigungseinstellanforderung an die MPU **15** ausgibt.

[0134] Danach gibt der Prozessorkern **12** eine Abrufanforderung an das Peripheriegerätesteuerungsprogramm **71** an die MPU **15** aus. Der Pfeil **L65** zeigt an, dass der Prozessorkern **12** den Abruf an das Peripheriegerät-Steuerprogramm **71** anfordert.

[0135] Da hier die Ausführungszugriffsberechtigung in der MPU **15** festgelegt ist, gibt die MPU **15** eine Benachrichtigung über die Ausführungszugriffsberechtigung an den Prozessorkern **12** aus. Nach Erwerb der Ausführungszugriffsberechtigung von der MPU **15** greift der Prozessorkern **12** auf das ROM **14** zu, holt das Peripheriegerät-Steuerprogramm **71** und

führt die Steuerung aus, um ein PWM-Signal an das Stellglied **4** auszugeben, wie durch den Pfeil **L66** angezeigt.

[0136] Außerdem gibt der Prozessorkern **13** eine Abrufanforderung an das Peripheriegerätesteuerungsprogramm **71** an die MPU **15** aus. Der Pfeil **L67** zeigt an, dass der Prozessorkern **13** den Abruf an das Peripheriegerät-Steuerprogramm **71** anfordert.

[0137] Da hier die Ausführungszugriffsberechtigung in der MPU **15** festgelegt wird, gibt die MPU **15** eine Benachrichtigung über die Ausführungszugriffsberechtigung an den Prozessorkern **13** aus. Nach Erhalt der Ausführungszugriffsberechtigung von der MPU **15** greift der Prozessorkern **13** auf das ROM **14** zu, holt das Peripheriegerät-Steuerprogramm **71** und führt die Steuerung aus, um ein PWM-Signal an das Stellglied **4** auszugeben, wie durch den Pfeil **L68** angezeigt.

[0138] Als nächstes wird ein zweites konkretes Beispiel für eine Execution Access Arbitration durch die MPU **15** beschrieben. Wie in **Fig. 16** gezeigt, legt der Prozessorkern **11** zunächst ein Ausführungszugriffsverbot fest, um einen Ausführungszugriff von den Prozessorkernen **12** und **13** auf das Peripheriegerätesteuerungsprogramm **71** zu verbieten. Der Pfeil **L71** zeigt an, dass der Prozessorkern **11** eine Verbotseinstellanforderung an die MPU **15** ausgibt.

[0139] Dann beginnt der Prozessorkern **11** mit der Ausführung eines Aktuatorsteuerungsprozesses zur Steuerung des Aktors **4**. Nach dem Start des Aktuatorsteuerungsprozesses gibt der Prozessorkern **12** eine Abrufanforderung an das Peripheriegerätesteuerungsprogramm **71** an die MPU **15** aus. Der Pfeil **L72** zeigt an, dass der Prozessorkern **12** die Abrufanforderung an das Peripheriegerätesteuerprogramm **71** anfordert.

[0140] Da hier das Ausführungszugriffsverbot in der MPU **15** festgelegt ist, gibt die MPU **15** eine Mitteilung über das Ausführungszugriffsverbot an den Prozessorkern **12** aus, wie durch den Pfeil **L73** angezeigt wird. Nach Erhalt der Ausführungszugriffsverbotsmeldung von der MPU **15** setzt der Prozessorkern **12** „Ausführungszugriffsanforderung vorhanden“ in die Handshake-Daten **62**, wie durch den Pfeil **L74** angezeigt.

[0141] Danach bestätigt der Prozessorkern **12** die Handshake-Daten **62**, wie durch das Quadrat **SQ7** und den Pfeil **L75** angezeigt, und wartet, bis „Ausführungszugriffsanforderung nicht vorhanden“ in den Handshake-Daten **62** eingestellt ist.

[0142] Andererseits wiederholt der Prozessorkern **11**, wenn die Ausführung der Aktor-Steuerungsverarbeitung gestartet wird, die durch die Pfeile **L76**, **L77**, **L78** und **L80** im Quadrat **SQ6** angezeigte Verarbeitung. Der Pfeil **L76** zeigt einen Prozess zur Diagnose des Aktors **4** an. Der Pfeil **L77** zeigt einen Prozess zum Abrufen des Peripheriegerät-Steuerprogramms **71** aus dem ROM **14** an. Der Pfeil **L78** ist ein Prozess zur Ausgabe eines Ausgabebefehlswerts an das Peripheriegerät **17**. Der Pfeil **L80** ist ein Prozess zum Bestätigen der Handshake-Daten **61**, **62**, **63**.

[0143] Wenn der Ausgangsbefehlswert vom Prozessorkern **11** erfasst wird, gibt das Peripheriegerät **17** ein PWM-Signal an das Stellglied **4** aus, wie durch den Pfeil **L79** angezeigt. Wenn dann „Ausführungszugriffsanforderung vorhanden“ in den Handshake-Daten **62** eingestellt ist, legt der Prozessorkern **11** eine Ausführungszugriffsberechtigung fest, um einen Ausführungszugriff von den Prozessorkernen **12** und **13** auf das Peripheriegerät-Steuerprogramm **71** zu erlauben. Der Pfeil **L81** zeigt an, dass der Prozessorkern **11** eine Berechtigungseinstellanforderung an die MPU **15** ausgibt.

[0144] Darüber hinaus setzt der Prozessorkern **11** „Ausführungszugriffsanforderung nicht vorhanden“ in den Handshake-Daten **62**, wie durch den Pfeil **L82** angezeigt. Wenn „Ausführungszugriffsanforderung nicht vorhanden“ (Abwesenheit der Ausführungszugriffsanforderung) in den Handshake-Daten **62** eingestellt ist, gibt der Prozessorkern **12** eine Abrufanforderung an das Peripheriegerät-Steuerprogramm **71** an die MPU **15** aus. Der Pfeil **L83** zeigt an, dass der Prozessorkern **12** die Abrufanforderung an das Peripheriegerät-Steuerprogramm **71** anfordert.

[0145] Da hier die Ausführungszugriffsberechtigung in der MPU **15** festgelegt ist, gibt die MPU **15** eine Benachrichtigung über die Ausführungszugriffsberechtigung an den Prozessorkern **12** aus. Nach Erwerb der Ausführungszugriffsberechtigung von der MPU **15** greift der Prozessorkern **12** auf das ROM **14** zu, holt das Peripheriegerät-Steuerprogramm **71** und gibt einen Ausgabebefehlswert an das Peripheriegerät **17** aus, wie durch den Pfeil **L84** angezeigt.

[0146] Nachdem der Ausgangsbefehlswert vom Prozessorkern **12** erfasst wurde, gibt das Peripheriegerät **17** ein PWM-Signal an das Stellglied **4** aus, wie durch den Pfeil **L85** angezeigt. Die wie oben beschrieben konfigurierte ECU **1** umfasst die Prozessorkerne **11**, **12** und **13**, das ROM **14** und die MPU **15**.

[0147] Das ROM **14** speichert das Peripheriegerät-Steuerprogramm **71**, auf das von jedem der Prozessorkerne **11**, **12** und **13** lesbar zugegriffen werden kann. Die MPU **15** steuert die Prozessorkerne **11**, **12**, **13**, so dass die Prozessorkerne **12**, **13** nicht auf das Peripheriegerätesteuerprogramm **71** zugreifen können, wenn der Prozessorkern **11** auf das Peripheriegerätesteuerprogramm **71** zugreift.

[0148] Wie oben beschrieben, steuert in der ECU **1** die MPU **15** die Prozessorkerne **11**, **12**, **13**, so dass die Prozessorkerne **12**, **13** nicht auf das Peripheriegerätesteuerprogramm **71** zugreifen können, wenn der Prozessorkern **11** auf das Peripheriegerätesteuerprogramm **71** zugreift. Das heißt, da das Steuergerät **1** die MPU **15** anstelle einer Semaphore verwendet, ist auf der Seite des Prozessorkerns, die auf das Programm zugreift, keine spezielle Schnittstelle erforderlich. Infolgedessen kann die ECU **1** aufgrund der ausschließlichen Steuerung zwischen den Prozessorkernen einen Anstieg des Overheads und eine Zunahme des Programmcodes unterdrücken.

[0149] Wenn die Verarbeitung zwischen den Prozessorkernen **11**, **12** und **13** nicht synchronisiert ist, greifen die Prozessorkerne **11**, **12** und **13** außerdem gleichzeitig auf das Peripheriegerät **17** zu, und es kommt zu Fehlfunktionen im Peripheriegerät **17**. Eine solche Fehlfunktion kann von der ECU **1** unterdrückt werden. Darüber hinaus kann das Steuergerät **1** eine schnelle Übertragung des Rechts zur Nutzung des Peripheriegeräts **17** zwischen den Prozessorkernen **11**, **12** und **13** realisieren, wodurch die Zuverlässigkeit und Reaktionsfähigkeit eines Systems mit dem Steuergerät **1** und dem Stellglied **4** verbessert wird.

[0150] Der Prozessorkern **11** setzt die MPU **15** auf ein Ausführungszugriffsverbot, das die Prozessorkerne **12** und **13** daran hindert, während eines Zeitraums, in dem der Prozessorkern **11** auf das Peripheriegerätesteuerungsprogramm **71** zugreift, einen Ausführungszugriff auf das Peripheriegerätesteuerungsprogramm **71** durchzuführen. Weiterhin setzt der Prozessorkern **11** die MPU **15** auf eine Ausführungszugriffsberechtigung, die es den Prozessorkernen **12** und **13** erlaubt, einen Ausführungszugriff auf das Peripheriegerätesteuerungsprogramm **71** während eines Zeitraums durchzuführen, in dem der Prozessorkern **11** nicht auf das Peripheriegerätesteuerungsprogramm **71** zugreift.

[0151] Wie oben beschrieben, verwendet der ECU **1** die MPU **15** zur Erkennung einer beabsichtigten Ausführungszugriffsanforderung, indem er absichtlich einen Ausführungszugriff von einem anderen Prozessorkern verbietet. Infolgedessen kann die ECU **1** den Overhead reduzieren, der bei jeder Verwendung einer Semaphore ständig erzeugt wird.

[0152] Im RAM **26** des Steuergeräts **1** werden die Handshake-Daten **62**, **63** gespeichert, die für das Handshake zwischen den Prozessorkernen **11**, **12**, **13** verwendet werden und auf die von jedem der Prozessorkerne **11**, **12**, **13** lesbar und aktualisierbar zugegriffen werden kann.

[0153] Die Prozessorkerne **12** und **13** geben jeweils innerhalb eines Zeitraums, in dem der Prozessorkern **11** auf das Peripheriegerätesteuerungsprogramm **71** zugreift, eine Abrufanforderung an das Peripheriegerätesteuerungsprogramm **71** an die MPU **15** aus. In einem solchen Fall gibt die MPU **15** eine Ausführungszugriffsverbotsmeldung an die Prozessorkerne **12** und **13** aus.

[0154] Die Prozessorkerne **12** und **13** erhalten jeweils eine Ausführungszugriffsverbotsmeldung von der MPU **15** und setzen „Ausführungszugriffsanforderung vorhanden“. Hier zeigt „Ausführungszugriffsanforderung vorhanden“ an, dass in den Handshaking-Daten **62** und **63**, die für das Handshaking zwischen dem Prozessorkern **11** und den Prozessorkernen **12** und **13** verwendet werden, eine Ausführungszugriffsanforderung vorhanden ist.

[0155] Die Handshake-Daten **62**, **63** wechseln von dem Zustand, in dem „Ausführungszugriffsanforderung vorhanden“ (Anwesenheit von „Ausführungszugriffsanforderung“) auf den Zustand, in dem „Ausführungszugriffsanforderung nicht vorhanden“ (Abwesenheit von „Ausführungszugriffsanforderung“) gesetzt ist. In diesem Fall gibt jeder der Prozessorkerne **12** und **13** eine Abrufanforderung an das Peripheriegerätesteuerprogramm **71** an die MPU **15** wieder aus.

[0156] Der Prozessorkern **11** bestätigt wiederholt die Handshake-Daten **62** und **63** jedes Mal, wenn der Prozess der Ausgabe des Ausgabebefehlswerts an das Peripheriegerät **17** innerhalb des Zeitraums abgeschlossen ist, in dem der Prozessorkern **11** auf das Peripheriegerät-Steuerprogramm **71** zugreift. Der Prozessorkern **11** bestimmt dann, ob in den Handshake-Daten **62**, **63** das Vorhandensein einer „Ausführungszugriffsanforderung“ eingestellt ist.

[0157] Der Prozessorkern **11** bestimmt, dass „Ausführungszugriffsanforderung vorhanden“ in den Handshake-Daten **62**, **63** gesetzt wird. In diesem Fall unterbricht der Prozessorkern **11** den Prozess des Abrufs des Peripheriegerät-Steuerprogramms **71** und der Ausgabe des Ausgabebefehlswerts an das Peripheriegerät **17** und setzt die MPU **15** auf die Ausführungszugriffsberechtigung.

[0158] Der Prozessorkern **11** bestimmt, dass in den Handshake-Daten **62**, **63** die „Leseanforderung vorhanden“ eingestellt ist. In diesem Fall legt der Prozessorkern **11** „Ausführungszugriffsanforderung nicht vorhanden“ fest, was anzeigt, dass in den Handshake-Daten **62**, **63** keine Ausführungszugriffsanforderung vorhanden ist.

[0159] Infolgedessen erkennt der Prozessorkern **11** die Ausführungszugriffsanforderung von den Prozessorkernen **12** und **13**. In diesem Fall unterbricht das Steuergerät **1** den Prozess des Abrufs des Peripheriegerät-Steuerprogramms **71** und der Ausgabe des Ausgabebefehlswerts an das Peripheriegerät **17** und veranlasst die Prozessorkerne **12** und **13**, den Aus-

führungszugriff auf das Peripheriegerät-Steuerprogramm **71** durchzuführen. Aus diesem Grund kann das Steuergerät **1** die Zeit verkürzen, die die Prozessorkerne **12** und **13** auf den Ausführungszugriff auf das Peripheriegerät-Steuerprogramm **71** warten, und kann die Verarbeitungseffizienz der Prozessorkerne **12** und **13** verbessern.

[0160] In der oben beschriebenen Ausführungsform entspricht das ROM **14** einer gemeinsam genutzten Datenspeichereinheit, das Peripheriegerät-Steuerprogramm **71** entspricht einer gemeinsam genutzten Dateneinheit und das RAM **26** entspricht einer Handshake-Speichereinheit. Das Ausführungszugriffsverbot entspricht einem Zugriffsverbot; die Ausführungszugriffserlaubnis entspricht einer Zugriffserlaubnis.

[0161] Der RAM **26** entspricht einer Handshake-Speichereinheit; eine Ausgabe der Ausführungszugriffsverbotsmeldung entspricht einem Speicherzugriffsverletzungsprozess; und „Ausführungszugriffsanforderung vorhanden“ entspricht einem Vorhandensein einer Zugriffsanforderung.

[0162] Ferner entspricht „Beendigung des Prozesses der Ausgabe des Ausgabebefehlswerts an das Peripheriegerät **17**“ einer Bestätigungsbedingung. „Der Prozess des Abrufens des Peripheriegerät-Steuerprogramms **71** und der Ausgabe des Ausgabebefehlswerts an das Peripheriegerät **17**“ ist ein Zugriffsprozess. Die „Ausführungszugriffsanforderung nicht vorhanden“ entspricht der Abwesenheit einer Zugriffsanforderung.

[Vierte Ausführungsform]

[0163] Im Folgenden wird eine vierte Ausführungsform der vorliegenden Offenbarung unter Bezugnahme auf die Zeichnungen beschrieben. Beachten Sie, dass in der vierten Ausführungsform Teile beschrieben werden, die sich von der dritten Ausführungsform unterscheiden. Gemeinsame Komponenten werden mit den gleichen Bezugszeichen beschrieben.

[0164] Die ECU **1** der vierten Ausführungsform unterscheidet sich von der dritten Ausführungsform dadurch, dass die in der zweiten Ausführungsform dargestellte Prioritätstabelle **TB2** im ROM **14** gespeichert ist. Als nächstes wird ein konkretes Beispiel für eine Execution Access Arbitration durch die MPU **15** der vierten Ausführungsform beschrieben.

[0165] Wie in **Fig. 17** dargestellt, legt der Prozessorkern **11** zunächst ein Ausführungszugriffsverbot fest, das den Ausführungszugriff von den Prozessorkernen **12** und **13** auf das Peripheriegerätesteuerungsprogramm **71** verbietet. Der Pfeil **L91** zeigt an, dass der Prozessorkern **11** eine Verbotseinstellanforderung an die MPU **15** ausgibt.

[0166] Dann beginnt der Prozessorkern **11** mit der Ausführung eines Aktuatorsteuerungsprozesses zur Steuerung des Aktors **4**. Nach dem Start des Aktuatorsteuerungsprozesses gibt der Prozessorkern **12** eine Abrufanforderung an das Peripheriegerätesteuerungsprogramm **71** an die MPU **15** aus. Der Pfeil **L92** zeigt an, dass der Prozessorkern **12** den Abruf an das Peripheriegerät-Steuerprogramm **71** angefordert hat.

[0167] Da hier das Ausführungszugriffsverbot in der MPU **15** festgelegt ist, gibt die MPU **15** eine Mitteilung über das Ausführungszugriffsverbot an den Prozessorkern **12** aus, wie durch den Pfeil **L93** angezeigt wird. Nach Erhalt der Mitteilung über das Ausführungszugriffsverbot von der MPU **15** setzt der Prozessorkern **12** „Ausführungszugriffsanforderung vorhanden“ in die Handshake-Daten **62**, wie durch den Pfeil **L94** angezeigt.

[0168] Danach bestätigt der Prozessorkern **12** die Handshake-Daten **62**, wie durch das Quadrat SQ9 und den Pfeil **L95** angezeigt, und wartet, bis in den Handshake-Daten **62** „Ausführungszugriffsanforderung nicht vorhanden“ eingestellt ist.

[0169] Ferner gibt der Prozessorkern **13** nach dem Start des Stellantriebs-Steuerungsprozesses eine Abrufanforderung an das Peripheriegerät-Steuerungsprogramm **71** an die MPU **15** aus. Der Pfeil **L96** zeigt an, dass der Prozessorkern **13** den Abruf an das Peripheriegerät-Steuerprogramm **71** angefordert hat.

[0170] Da hier das Ausführungszugriffsverbot in der MPU **15** festgelegt ist, gibt die MPU **15** eine Mitteilung über das Ausführungszugriffsverbot an den Prozessorkern **13** aus, wie durch den Pfeil **L97** angezeigt wird. Nach Erhalt der Ausführungszugriffsverbotsmeldung von der MPU **15** setzt der Prozessorkern **13** „Ausführungszugriffsanforderung vorhanden“ in die Handshake-Daten **63**, wie durch den Pfeil **L98** angezeigt.

[0171] Danach bestätigt der Prozessorkern **13** die Handshake-Daten **63**, wie durch das Quadrat SQ 10 und den Pfeil **L99** angezeigt, und wartet, bis in den Handshake-Daten **63** „Ausführungszugriffsanforderung nicht vorhanden“ eingestellt ist.

[0172] Andererseits wiederholt der Prozessorkern **11**, wenn die Ausführung der Aktor-Steuerungsverarbeitung gestartet wird, die Verarbeitung, die durch die Pfeile **L100**, **L101** und **L102** im Quadrat SQ8 angezeigt wird. Der Pfeil **L100** zeigt einen Vorgang zum Abrufen des Peripheriegerät-Steuerprogramms **71** aus dem ROM **14** an. Der Pfeil **L101** ist ein Prozess der Ausgabe eines PWM-Signals an das Stellglied **4**. Der Pfeil **L102** ist ein Prozess zum Bestätigen der Handshake-Daten **61**, **62**, **63**.

[0173] Wenn dann „Ausführungszugriffsanforderung vorhanden“ in den Handshake-Daten **62** eingestellt ist, wird auf die im ROM **14** gespeicherte Prioritätstabelle **TB2** verwiesen, wie durch den Pfeil **L103** angezeigt.

[0174] Als nächstes legt der Prozessorkern **11** eine Ausführungszugriffsberechtigung fest, um einen Ausführungszugriff vom Prozessorkern **12** auf das Peripheriegerätesteuerungsprogramm **71** zu ermöglichen. Der Pfeil **L104** zeigt an, dass der Prozessorkern **11** eine Berechtigungseinstellanforderung an die MPU **15** ausgibt.

[0175] Darüber hinaus setzt der Prozessorkern **11** „Ausführungszugriffsanforderung nicht vorhanden“ in den Handshake-Daten **62**, wie durch den Pfeil **L105** angezeigt. Wenn „Ausführungszugriffsanforderung nicht vorhanden“ (Abwesenheit der Ausführungszugriffsanforderung) in den Handshake-Daten **62** eingestellt ist, gibt der Prozessorkern **12** eine Abrufanforderung an das Peripheriegerät-Steuerprogramm **71** an die MPU **15** aus. Der Pfeil **L106** zeigt an, dass der Prozessorkern **12** eine Fetch-Anforderung an das Peripheriegerät-Steuerprogramm **71** ausgibt.

[0176] Da hier die Ausführungszugriffsberechtigung in der MPU **15** festgelegt ist, gibt die MPU **15** eine Benachrichtigung über die Ausführungszugriffsberechtigung an den Prozessorkern **12** aus. Nach Erwerb der Ausführungszugriffsberechtigung von der MPU **15** greift der Prozessorkern **12** auf das ROM **14** zu, holt das Peripheriegerät-Steuerprogramm **71** und gibt ein PWM-Signal an den Stellantrieb **4** aus, wie durch den Pfeil **L107** angezeigt.

[0177] Dann verweist der Prozessorkern **12** auf die im ROM **14** gespeicherte Prioritätstabelle **TB2**, wie durch den Pfeil **L108** angezeigt. Als nächstes setzt der Prozessorkern **12** eine Ausführungszugriffsberechtigung, um einen Ausführungszugriff vom Prozessorkern **13** auf das Peripheriegerätesteuerungsprogramm **71** zu erlauben. Der Pfeil **L109** zeigt an, dass der Prozessorkern **11** eine Berechtigungseinstellanforderung an die MPU **15** ausgibt.

[0178] Darüber hinaus setzt der Prozessorkern **12** „Ausführungszugriffsanforderung nicht vorhanden“ in den Handshake-Daten **63**, wie durch den Pfeil **L110** angezeigt. Wenn „Ausführungszugriffsanforderung nicht vorhanden“ (Abwesenheit der Ausführungszugriffsanforderung) in den Handshake-Daten **63** eingestellt ist, gibt der Prozessorkern **13** eine Abrufanforderung an das Peripheriegerät-Steuerprogramm **71** an die MPU **15** aus. Der Pfeil **L111** zeigt an, dass der Prozessorkern **13** den Abruf an das Peripheriegerät-Steuerprogramm **71** angefordert hat.

[0179] Da hier die Ausführungszugriffsberechtigung in der MPU **15** festgelegt wird, gibt die MPU **15** eine Benachrichtigung über die Ausführungszugriffsberechtigung an den Prozessorkern **13** aus. Nach Erhalt der Ausführungszugriffsberechtigung von der MPU **15** greift der Prozessorkern **13** auf das ROM **14** zu, holt das Peripheriegerät-Steuerprogramm **71** ab und gibt ein PWM-Signal an das Stellglied **4** aus, wie durch den Pfeil **L112** angezeigt.

[0180] Die wie oben beschrieben konfigurierte ECU **1** enthält die Prioritätstabelle **TB2**, in der die Priorität für jeden der Prozessorkerne **11**, **12** und **13** festgelegt ist. Dann setzt der Prozessorkern **11** auf der Grundlage der Prioritätstabelle **TB2** die MPU **15** auf eine Ausführungszugriffsberechtigung für den Prozessorkern **12**, der die höchste Priorität unter den Prozessorkernen **12** und **13** hat, die die Abrufanforderung an die MPU **15** ausgeben.

[0181] Ferner setzt der Prozessorkern **11** „Ausführungszugriffsanforderung nicht vorhanden“ in den Handshake-Daten **62**, die für den Handshake zwischen dem Prozessorkern **11** und dem Prozessorkern **12** verwendet werden.

[0182] So kann das Steuergerät **1** den Prozessorkern **12** mit der höheren Priorität veranlassen, auf das Peripheriegerätesteuerprogramm **71** zuzugreifen, bevor der Prozessorkern **13** die niedrigere Priorität hat. Auf diese Weise kann das Steuergerät **1** das Auftreten einer Situation unterdrücken, in der die Ausführung des Prozesses mit der höheren Priorität aufgrund der Ausführung des Prozesses mit der niedrigeren Priorität verzögert wird. Die Reaktionsfähigkeit der ECU **1** kann so verbessert werden.

[0183] Wie oben beschrieben, sind die Ausführungsformen der vorliegenden Offenbarung beschrieben worden, aber die vorliegende Offenbarung ist nicht auf die oben genannten Ausführungsformen beschränkt und kann mit verschiedenen Modifikationen umgesetzt werden. Die ECU **1** und ihre Techniken gemäß der vorliegenden Bekanntmachung können durch einen oder mehrere speziell dafür vorgesehene Computer implementiert werden. Ein solcher Computer mit besonderer Zweckbestimmung kann (i) durch Konfiguration (a) eines Prozessors und eines Speichers, der so programmiert ist, dass er eine oder mehrere durch ein Computerprogramm auszuführende Funktionen ausführt, oder (ii) durch Konfiguration (b) eines Prozessors, der eine oder mehrere dedizierte Hardware-Logikschaltungen enthält, oder (iii) durch Konfiguration einer Kombination aus (a) einem Prozessor und einem Speicher, der so programmiert ist, dass er eine oder mehrere durch ein Computerprogramm auszuführende Funktionen ausführt, und (b) einem Prozessor, der eine oder mehrere dedizierte Hardware-Logikschaltungen enthält, bereitgestellt werden. Ferner kann das Computerprogramm in einem computerlesbaren, nicht vorübergehenden materiellen Speichermedium als von einem Compu-

ter auszuführende Befehle gespeichert werden. Die Technik zur Realisierung der Funktionen jeder in der ECU **1** enthaltenen Einheit muss nicht notwendigerweise Software enthalten, und alle Funktionen können unter Verwendung einer oder mehrerer Hardwareschaltungen realisiert werden.

[0184] Eine Vielzahl von Funktionen einer Komponente in den oben beschriebenen Ausführungsformen kann durch eine Vielzahl von Komponenten realisiert werden, oder eine Funktion einer Komponente kann durch eine Vielzahl von Komponenten realisiert werden. Zusätzlich kann eine Vielzahl von Funktionen einer Vielzahl von Komponenten durch eine Komponente realisiert werden, oder eine Funktion, die durch eine Vielzahl von Komponenten realisiert wird, kann durch eine Komponente realisiert werden. Ferner kann ein Teil der Konfiguration der obigen Ausführungsformen weggelassen werden. Ferner kann zumindest ein Teil der Konfiguration der oben beschriebenen Ausführungsform zu der Konfiguration einer anderen oben beschriebenen Ausführungsform hinzugefügt oder durch diese ersetzt werden.

[0185] Die vorliegende Offenbarung kann zusätzlich zu der oben beschriebenen ECU **1** in verschiedenen Formen umgesetzt werden, wie z.B. ein System, das die ECU **1** als Komponente enthält, ein Programm, das bewirkt, dass ein Computer als ECU **1** funktioniert, ein nicht vorübergehendes materielles Speichermedium mit einem Halbleiterspeicher, der das Programm speichert, ein Zugriffssteuerungsverfahren.

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

- WO 2017/056725 A1 [0003]

Patentansprüche

1. Elektronische Steuereinheit, umfassend:
eine Vielzahl von Prozessorkernen (11, 12, 13), wobei einer der Vielzahl von Prozessorkernen als ein spezifischer Prozessorkern (11) definiert ist, während mindestens ein Prozessorkern der Vielzahl von Prozessorkernen, der nicht der spezifische Prozessorkern ist, als ein anderer Prozessorkern (12, 13) definiert ist;
eine gemeinsam genutzte Datenspeichereinheit (21, 14), die konfiguriert ist, gemeinsam genutzte Daten zu speichern, auf die von jedem der Vielzahl von Prozessorkernen lesbar zugegriffen werden kann; und
eine Speicherschutzeinheit (15), die konfiguriert ist, die Vielzahl von Prozessoren oder Prozessorkernen so zu steuern, dass der andere Prozessorkern deaktiviert wird, um auf die gemeinsam genutzten Daten zuzugreifen, wenn der spezifische Prozessorkern auf die gemeinsam genutzten Daten zugreift.

2. Elektronische Steuereinheit nach Anspruch 1, wobei
der spezifische Prozessorkern enthält:
eine Verbotseinstelleinheit (S110, S410), die konfiguriert ist, die Speicherschutzeinheit auf ein Zugriffsverbot einzustellen, das dem anderen Prozessorkern den Zugriff auf die gemeinsam genutzten Daten während eines Zeitraums verbietet, in welchem der spezifische Prozessorkern auf die gemeinsam genutzten Daten zugreift; und
eine Berechtigungseinstelleinheit (S180, S480), die konfiguriert ist, die Speicherschutzeinheit auf eine Zugriffsberechtigung einzustellen, die dem anderen Prozessorkern den Zugriff auf die gemeinsam genutzten Daten während eines Zeitraums erlaubt, in welchem der spezifische Prozessorkern nicht auf die gemeinsam genutzten Daten zugreift.

3. Elektronische Steuereinheit nach Anspruch 2, ferner umfassend:
eine Handshake-Speichereinheit (21, 26), die konfiguriert ist, Handshake-Daten (41, 51, 62, 63) zu speichern, welche zum Handshaking zwischen der Vielzahl von Prozessorkernen verwendet werden und welche von jedem der Vielzahl von Prozessorkernen gelesen und aktualisiert werden können,
wobei die Speicherschutzeinheit konfiguriert ist, einen Speicherzugriffsverletzungsprozess auf dem anderen Prozessorkern durchzuführen, wenn der andere Prozessorkern versucht, auf die gemeinsam genutzten Daten innerhalb des Zeitraums zuzugreifen, in welchem der spezifische Prozessorkern auf die gemeinsam genutzten Daten zugreift,
wobei der andere Prozessorkern enthält:
eine Anforderungsanwesenheitseinstelleinheit (S310, S630), die konfiguriert ist, eine Zugriffsanforderungsanwesenheit einzustellen, die anzeigt, dass eine Zugriffsanforderung in Verletzungsziel-Handshake-Daten vorhanden ist, welche zum Handshake zwischen dem spezifischen Prozessorkern und dem anderen Prozessorkern verwendet wird, wenn der Speicherzugriffsverletzungsprozess durchgeführt wird; und
eine Wiederzugriffseinheit (S320, S330, S640), die konfiguriert ist, einen Zugriff auf die gemeinsam genutzten Daten erneut zu versuchen, wenn sich die Verletzungsziel-Handshake-Daten von einem Zustand, in welchem die Zugriffsanforderungsanwesenheit eingestellt ist, in einen Zustand ändert, in welchem eine Zugriffsanforderungsabwesenheit, die anzeigt, dass keine Zugriffsanforderung vorliegt, eingestellt ist,
wobei der spezifische Prozessorkern enthält:
eine Anforderungsbestimmungseinheit (S150, S460), die konfiguriert ist, zu bestimmen, ob die Zugriffsanforderungsanwesenheit in den Verletzungsziel-Handshake-Daten eingestellt ist, indem die Verletzungsziel-Handshake-Daten jedes Mal wiederholt bestätigt werden, wenn eine voreingestellte Bestätigungsbedingung innerhalb des Zeitraums erfüllt ist, in welchem der spezifische Prozessorkern auf die gemeinsam genutzten Daten zugreift;
eine Unterbrechungseinheit (S150, S460, S490, S500), die konfiguriert ist, einen Zugriffsprozess für den spezifischen Prozessorkern zu unterbrechen, um auf die gemeinsam genutzten Daten zuzugreifen, und die Berechtigungseinstelleinheit veranlasst, die Speicherschutzeinheit auf die Zugriffsberechtigung einzustellen, wenn die Anforderungsbestimmungseinheit bestimmt, dass die Zugriffsanforderungsanwesenheit in den Verletzungsziel-Handshake-Daten eingestellt ist; und
eine Anforderungsabwesenheitseinstelleinheit (S190, S510), die konfiguriert ist, eine Zugriffsanforderungsabwesenheit auf die Verletzungsziel-Handshake-Daten einzustellen, wenn die Anforderungsbestimmungseinheit bestimmt, dass die Zugriffsanforderungsanwesenheit in den Verletzungsziel-Handshake-Daten eingestellt ist.

4. Elektronische Steuereinheit nach einem der Ansprüche 1 bis 3, wobei ein Prozess, in welchem der spezifische Prozessorkern auf die gemeinsam genutzten Daten zugreift, ein Test des Lesens von Daten aus den gemeinsam genutzten Daten und des Schreibens von Daten in die gemeinsam genutzten Daten ist.

5. Elektronische Steuereinheit nach Anspruch 3, ferner umfassend:
eine Prioritätstabelle (TB2), in welcher jeweilige Prioritäten für die Vielzahl von Prozessorkernen eingestellt sind,
wobei:
basierend auf der Prioritätstabelle, die Unterbrechungseinheit (S460, S490, S500) konfiguriert ist, die Berechtigungseinstelleinheit zu veranlassen, die Speicherschutzeinheit auf eine Zugriffsberechtigung auf einen Prozessorkern hoher Priorität einzustellen,

wobei der andere Prozessorkern, der unter der Vielzahl anderer Prozessorkerne, die versucht haben, auf die gemeinsam genutzten Daten zuzugreifen, die höchste Priorität hat; und

die Zugriffsanforderungsabwesenheitseinstelleinheit (S510) konfiguriert ist, die Zugriffsanforderungsabwesenheit auf die Verletzungsziel-Handshake-Daten einzustellen, die zum Handshake zwischen dem spezifischen Prozessorkern und dem Prozessorkern hoher Priorität verwendet werden.

Es folgen 16 Seiten Zeichnungen

Anhängende Zeichnungen

FIG. 1

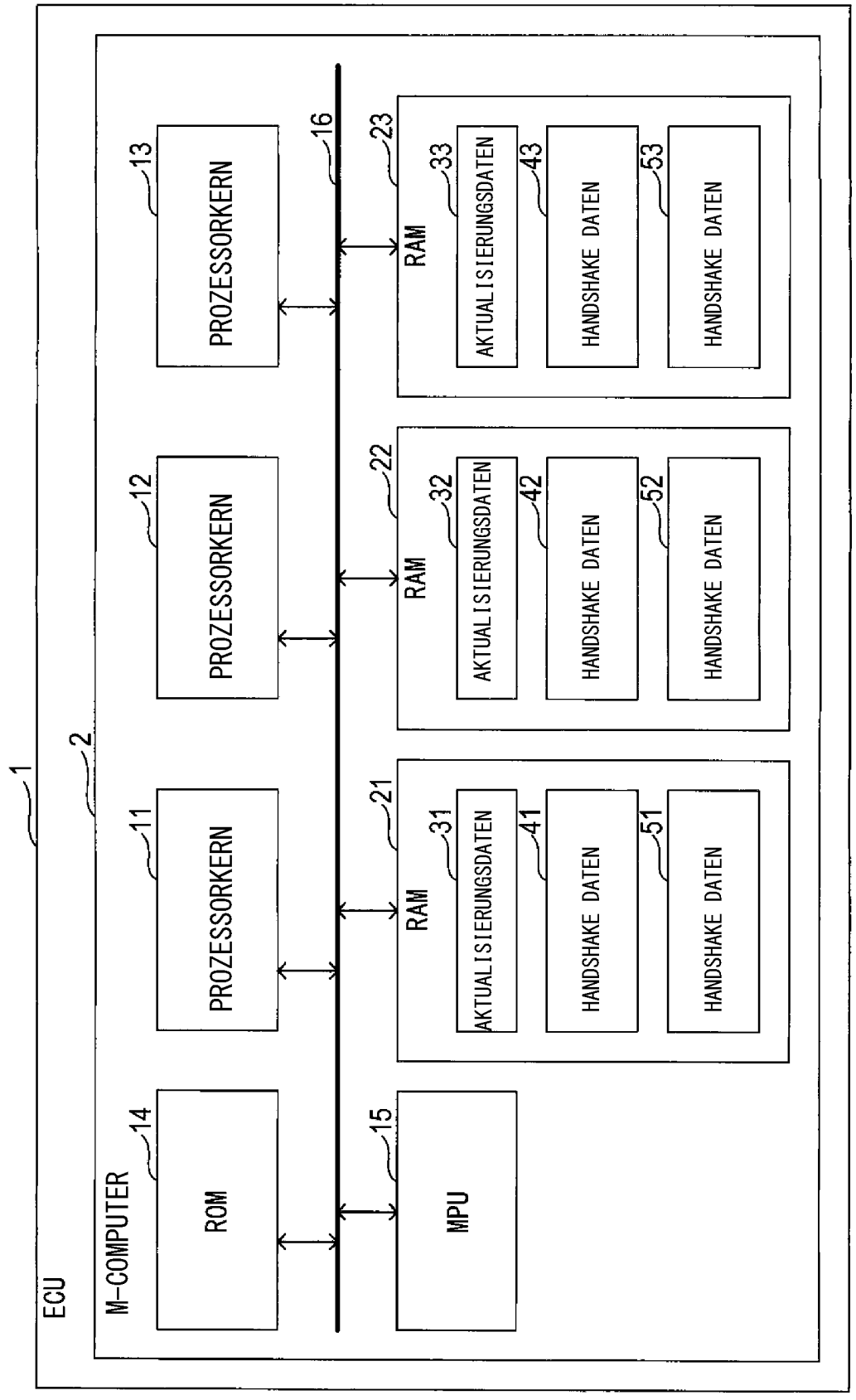


FIG. 2

TB1

1	NO. 1 BLOCK ADDRESSBEREICH
2	NO. 2 BLOCK ADDRESSBEREICH
3	NO. 3 BLOCK ADDRESSBEREICH
4	NO. 4 BLOCK ADDRESSBEREICH
5	NO. 5 BLOCK ADDRESSBEREICH

FIG. 3

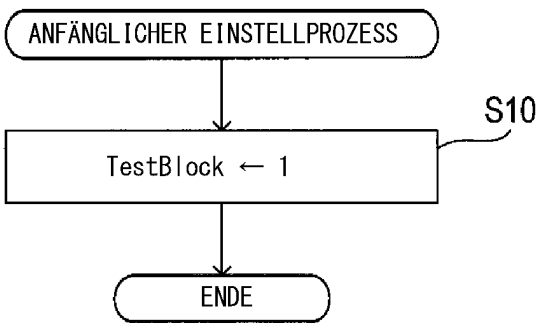


FIG. 4

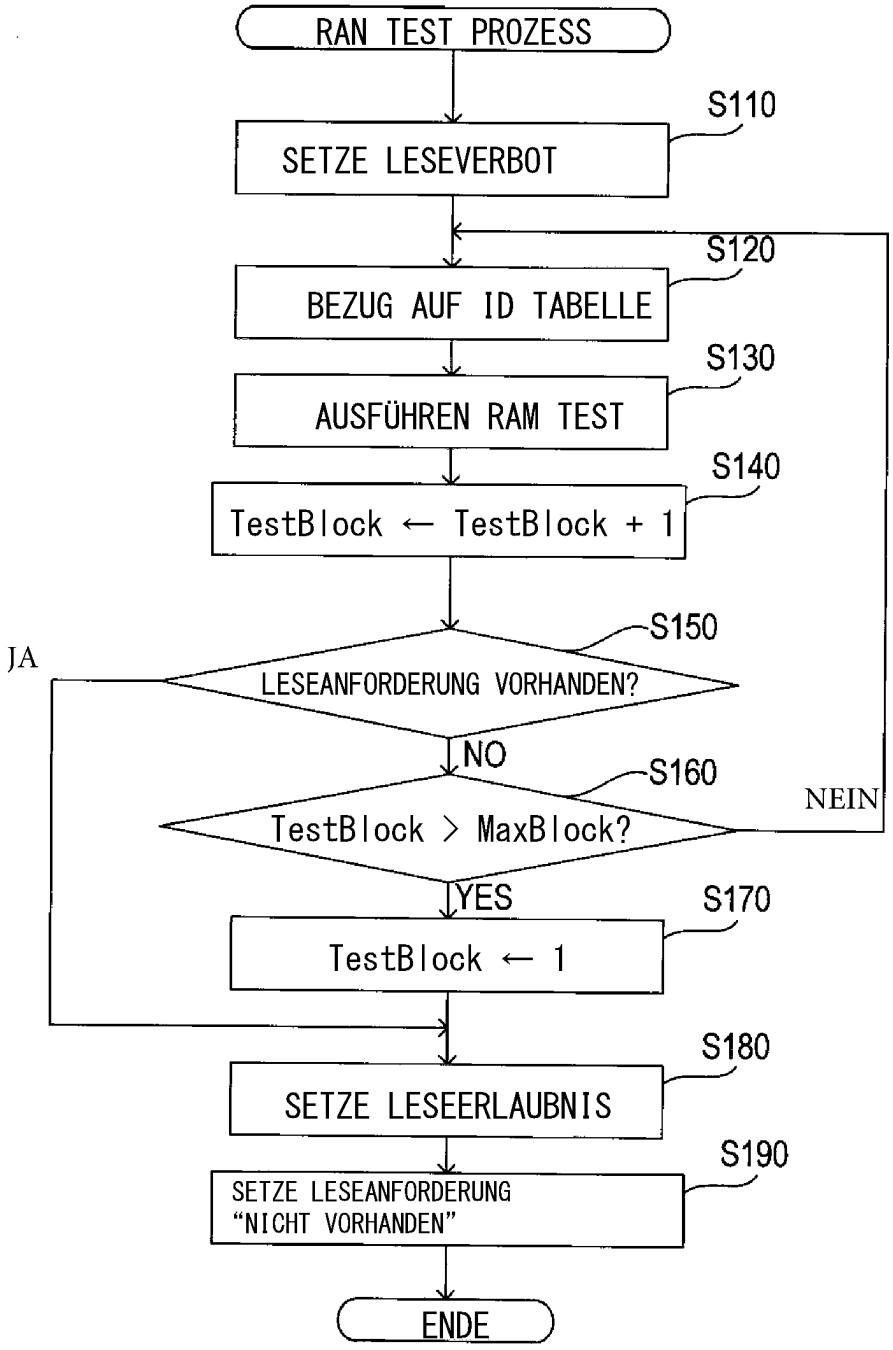

RAN TEST PROZESS
S110
SETZE LESEVERBOT
S120
BEZUG AUF ID TABELLE
S130
AUSFÜHREN RAM TEST
S140
TestBlock ← TestBlock + 1
S150
JA
LESEANFORDERUNG VORHANDEN?
NO
S160
NEIN
TestBlock > MaxBlock?
YES
S170
TestBlock ← 1
S180
SETZE LESEERLAUBNIS
S190
SETZE LESEANFORDERUNG
"NICHT VORHANDEN"
ENDE

FIG. 5

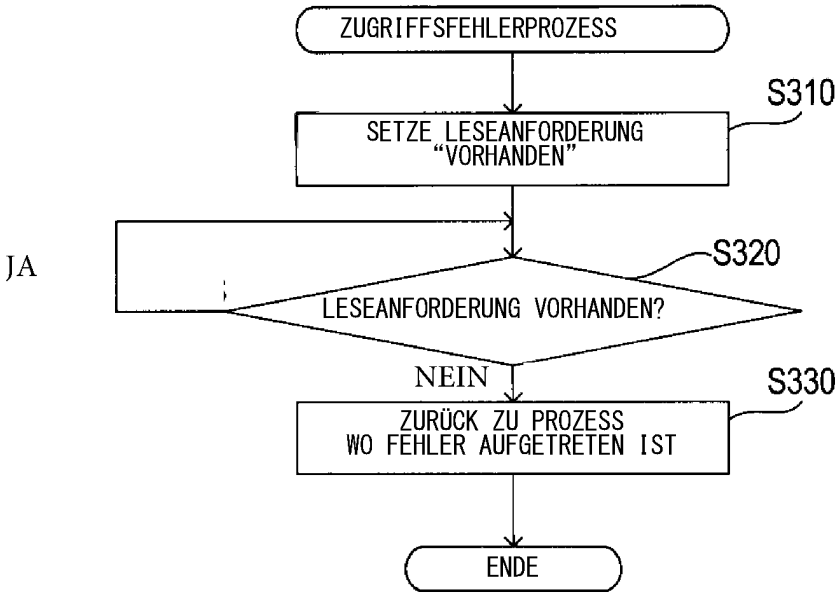

ZUGRIFFSFEHLERPROZESS
S310
SETZE LESEANFORDERUNG "VORHANDEN"
JA
S320
LESEANFORDERUNG VORHANDEN?
NEIN
S330
ZURÜCK ZU PROZESS WO FEHLER AUFGETRETEN IST
ENDE

FIG. 6

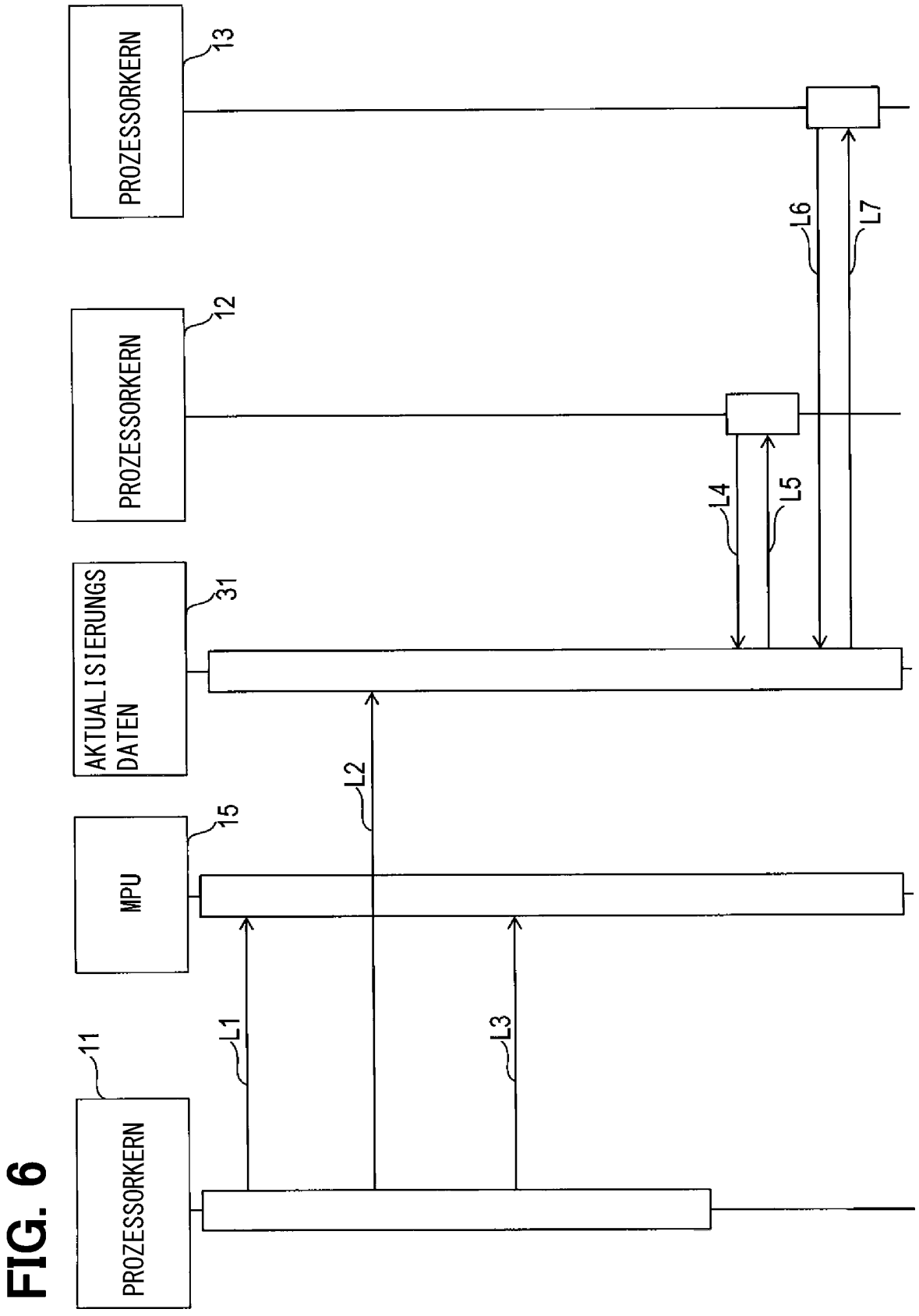

PROZESSORKERN
13
PROZESSORKERN
12
AKTUALISIERUNGS DATEN
31
MPU
15
PROZESSORKERN
11
L1
L2
L3
L4
L5
L6
L7

FIG. 7

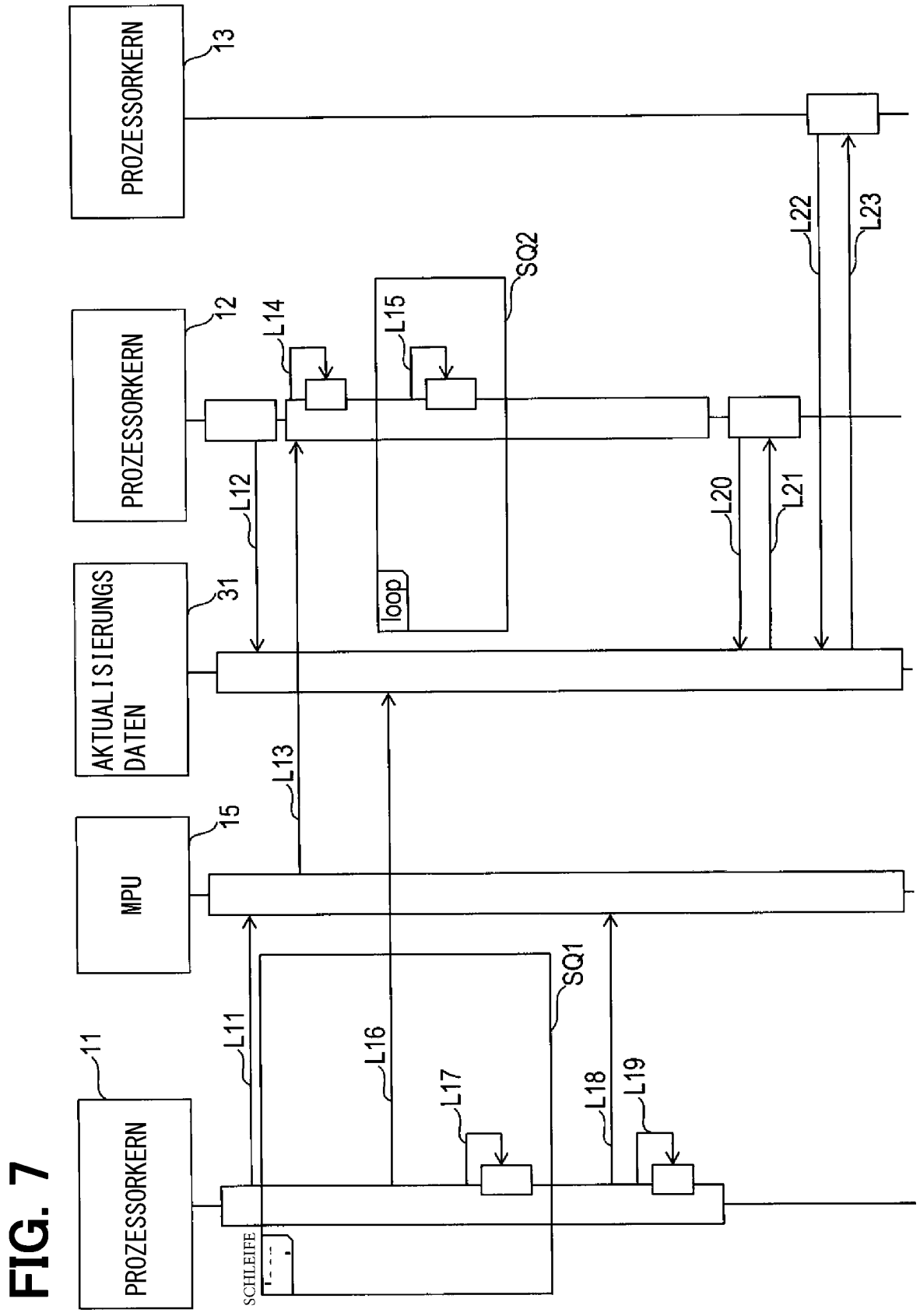

PROZESSORKERN
13
PROZESSORKERN
12
AKTUALISIERUNGS DATEN
31
MPU
15
PROZESSORKERN
11
L11
L12
L13
L14
L15
L16
L17
L18
L19
L20
L21
L22
L23
SQ1
SQ2
loop
SCHLEIFE

FIG. 8

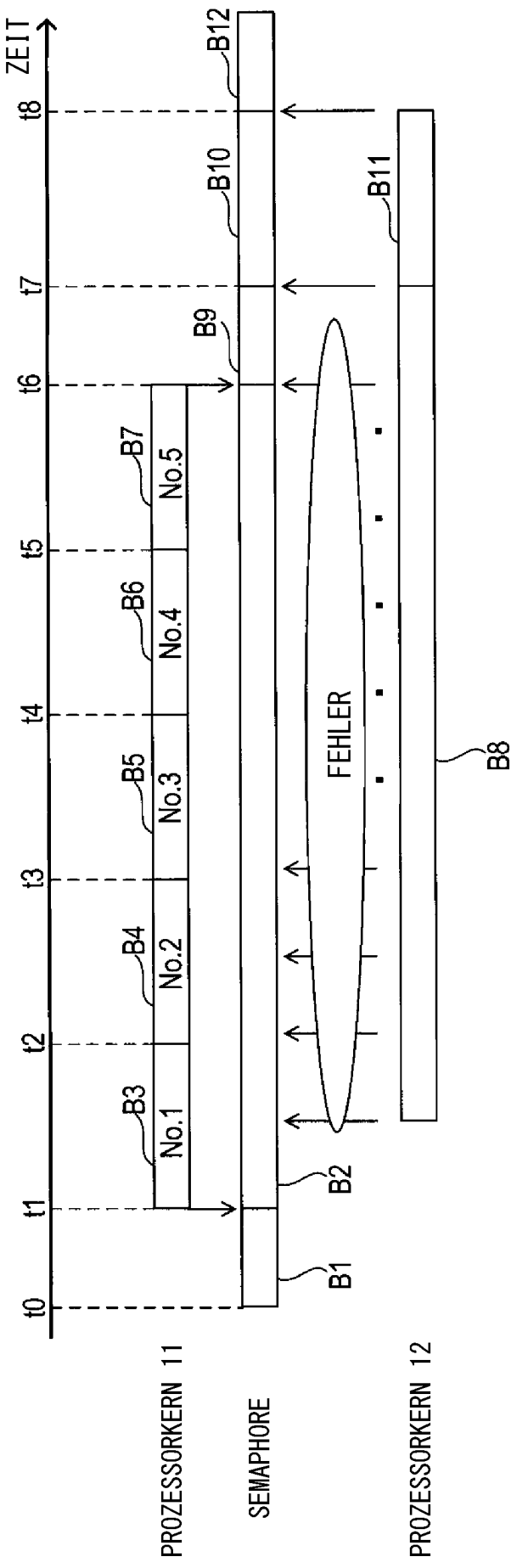

ZEIT
t0
t1
t2
t3
t4
t5
t6
t7
t8
B1
B2
B3
No.1
B4
No.2
B5
No.3
B6
No.4
B7
No.5
B8
B9
B10
B11
B12
FEHLER
PROZESSORKERN 11
SEMAPHORE
PROZESSORKERN 12

FIG. 9

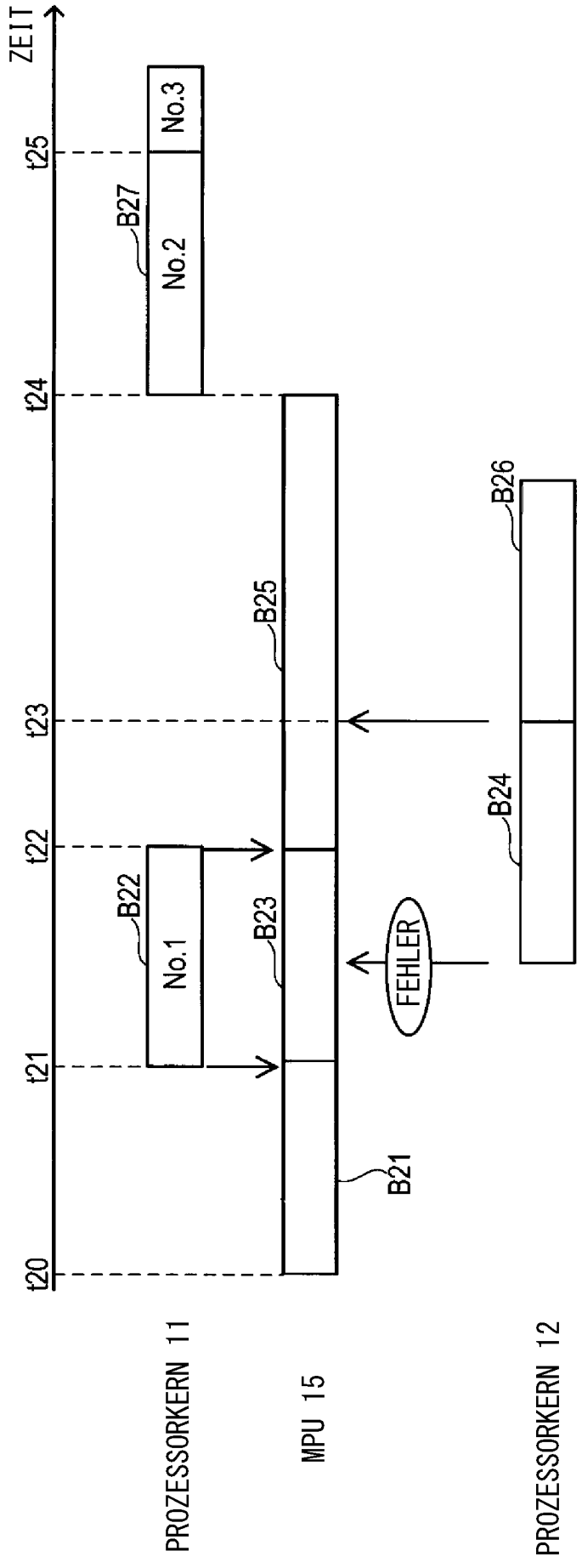

ZEIT
t20
t21
t22
t23
t24
t25
B21
B22
No.1
B23
B24
B25
B26
B27
No.2
No.3
FEHLER
PROZESSORKERN 11
MPU 15
PROZESSORKERN 12

FIG. 10

TB2

PROZESSORKERN NO.	PRIORITÄT
1	HOCH
2	MITTEL
3	NIEDRIG

FIG. 11

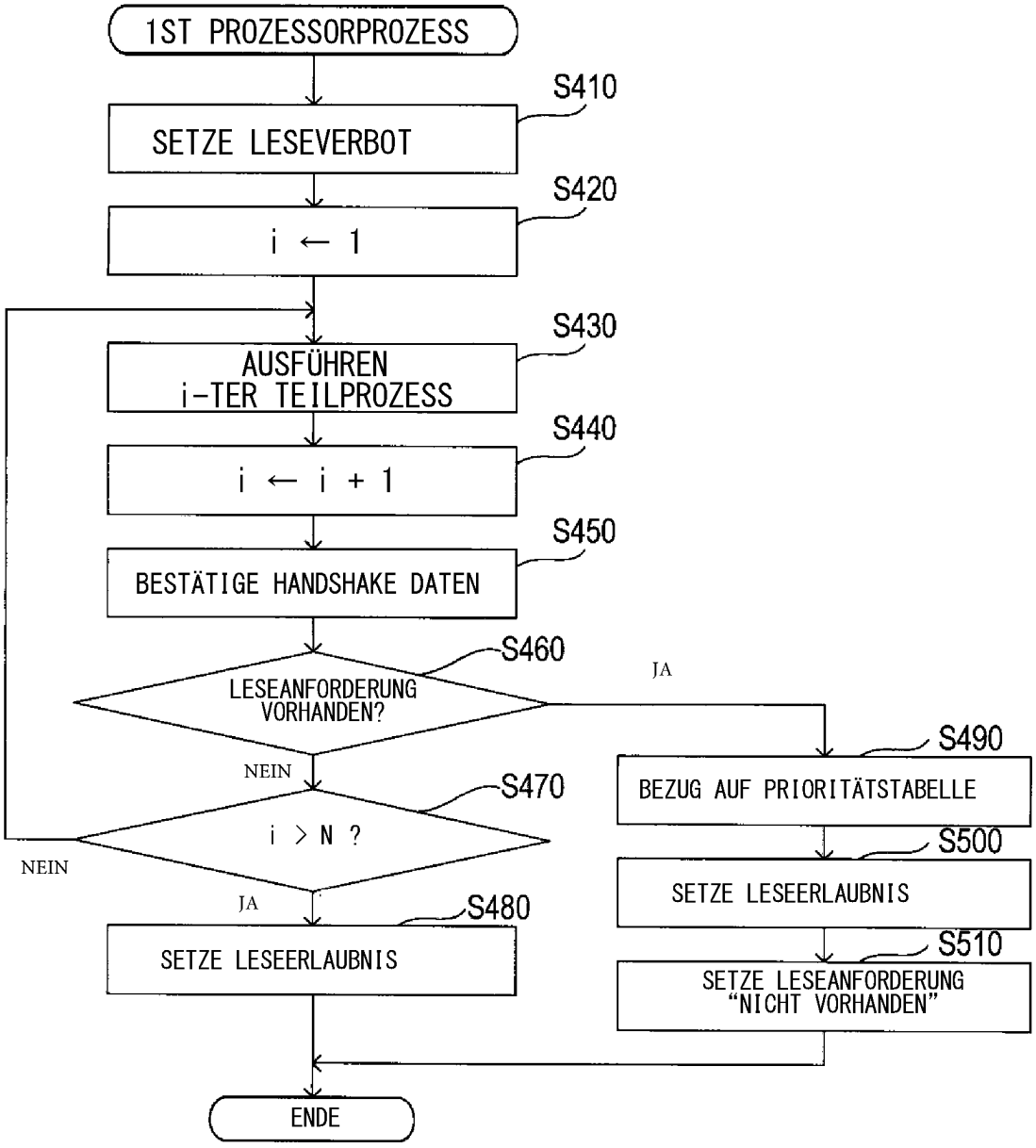

1ST PROZESSORPROZESS
S410
SETZE LESEVERBOT
S420
i ← 1
S430
AUSFÜHREN
i-TER TEILPROZESS
S440
i ← i + 1
S450
BESTÄTIGE HANDSHAKE DATEN
S460
LESEANFORDERUNG
VORHANDEN?
JA
NEIN
S470
i > N ?
NEIN
JA
S480
SETZE LESEERLAUBNIS
S490
BEZUG AUF PRIORITÄTSTABELLE
S500
SETZE LESEERLAUBNIS
S510
SETZE LESEANFORDERUNG
"NICHT VORHANDEN"
ENDE

FIG. 12

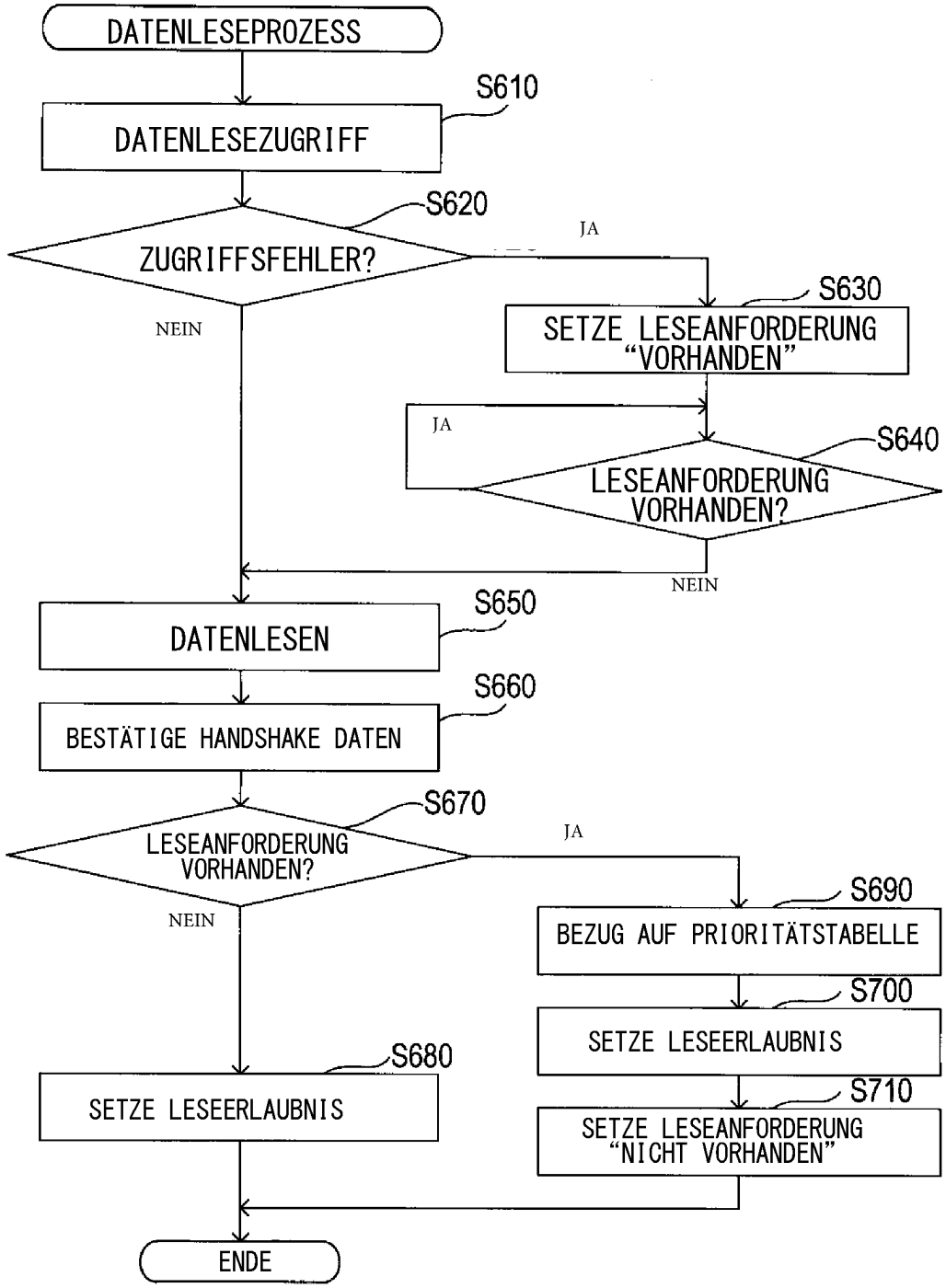

DATENLESEPROZESS
S610
DATENLESEZUGRIFF
S620
ZUGRIFFSFEHLER?
JA
NEIN
S630
SETZE LESEANFORDERUNG "VORHANDEN"
JA
S640
LESEANFORDERUNG VORHANDEN?
NEIN
S650
DATENLESEN
S660
BESTÄTIGE HANDSHAKE DATEN
S670
LESEANFORDERUNG VORHANDEN?
JA
NEIN
S690
BEZUG AUF PRIORITÄTSTABELLE
S700
SETZE LESEERLAUBNIS
S710
SETZE LESEANFORDERUNG "NICHT VORHANDEN"
S680
SETZE LESEERLAUBNIS
ENDE

FIG. 13

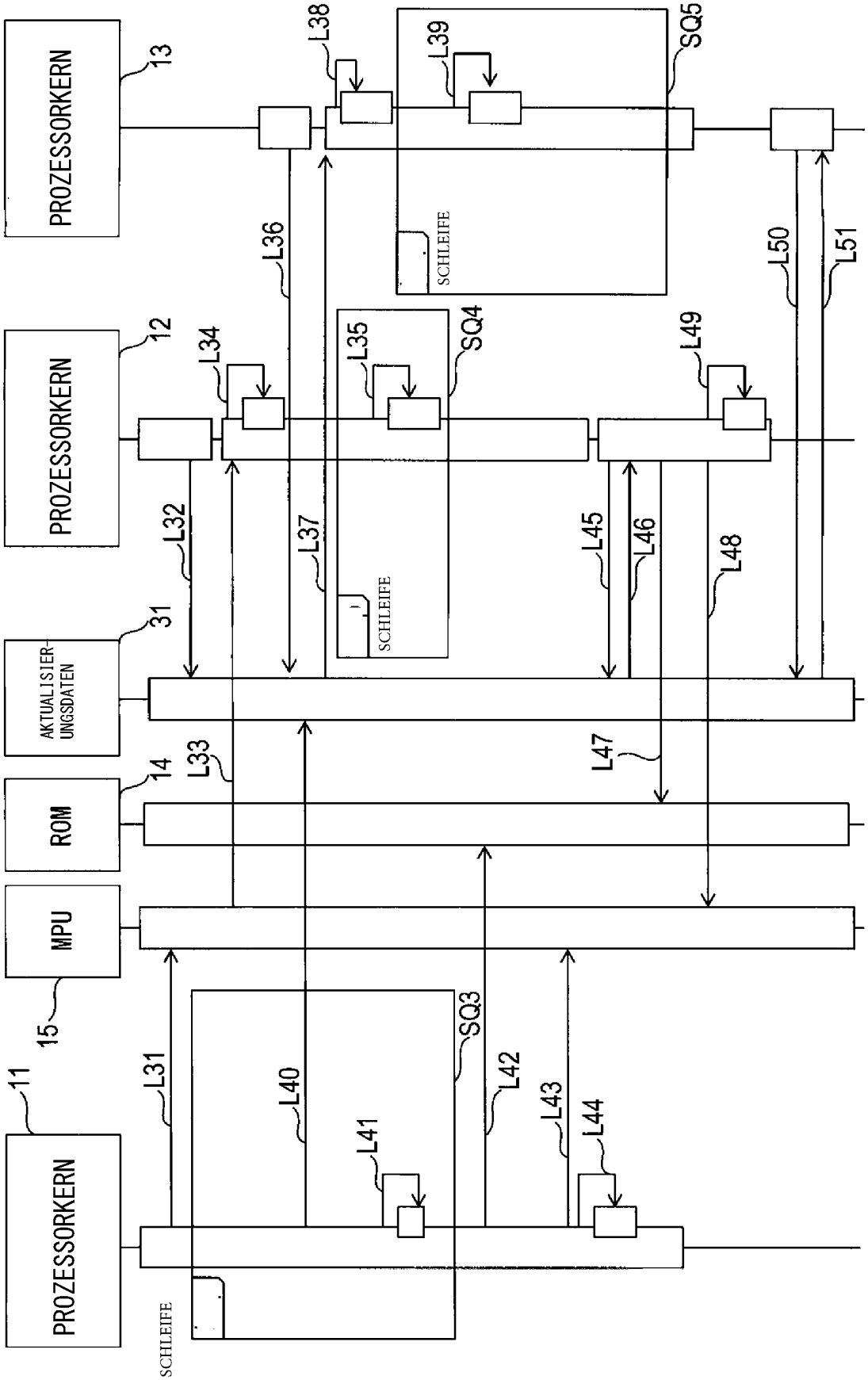

PROZESSORKERN
13
PROZESSORKERN
12
AKTUALISIER-UNGSDATEN
31
ROM
14
MPU
15
PROZESSORKERN
11
SCHLEIFE
SCHLEIFE
SCHLEIFE
L31
L32
L33
L34
L35
L36
L37
L38
L39
L40
L41
L42
L43
L44
L45
L46
L47
L48
L49
L50
L51
SQ3
SQ4
SQ5

FIG. 14

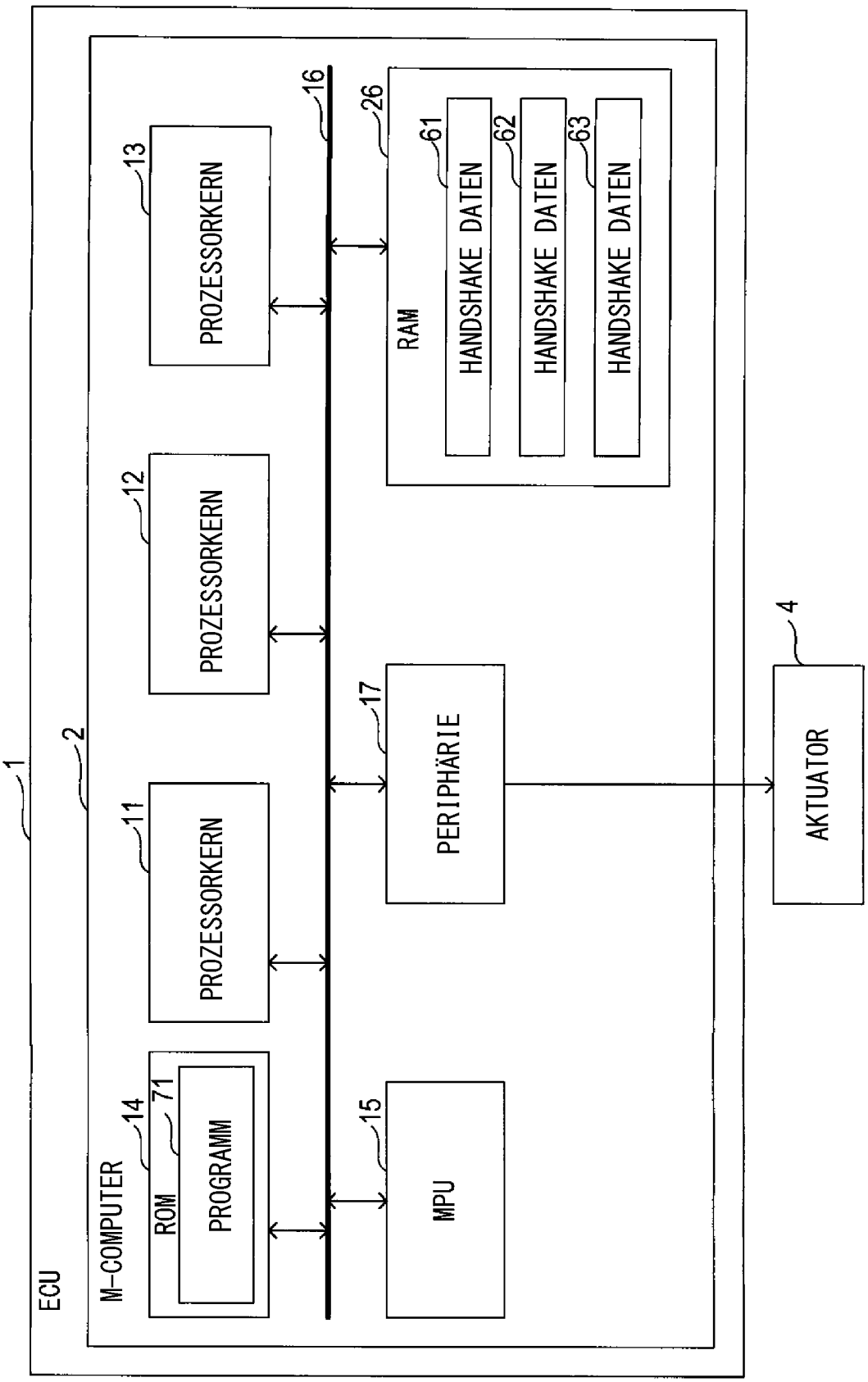

1
ECU
2
M-COMPUTER
14
ROM
71
PROGRAMM
11
PROZESSORKERN
12
PROZESSORKERN
13
PROZESSORKERN
15
MPU
16
17
PERIPHÄRIE
26
RAM
61
HANDSHAKE DATEN
62
HANDSHAKE DATEN
63
HANDSHAKE DATEN
4
AKTUATOR

FIG. 15

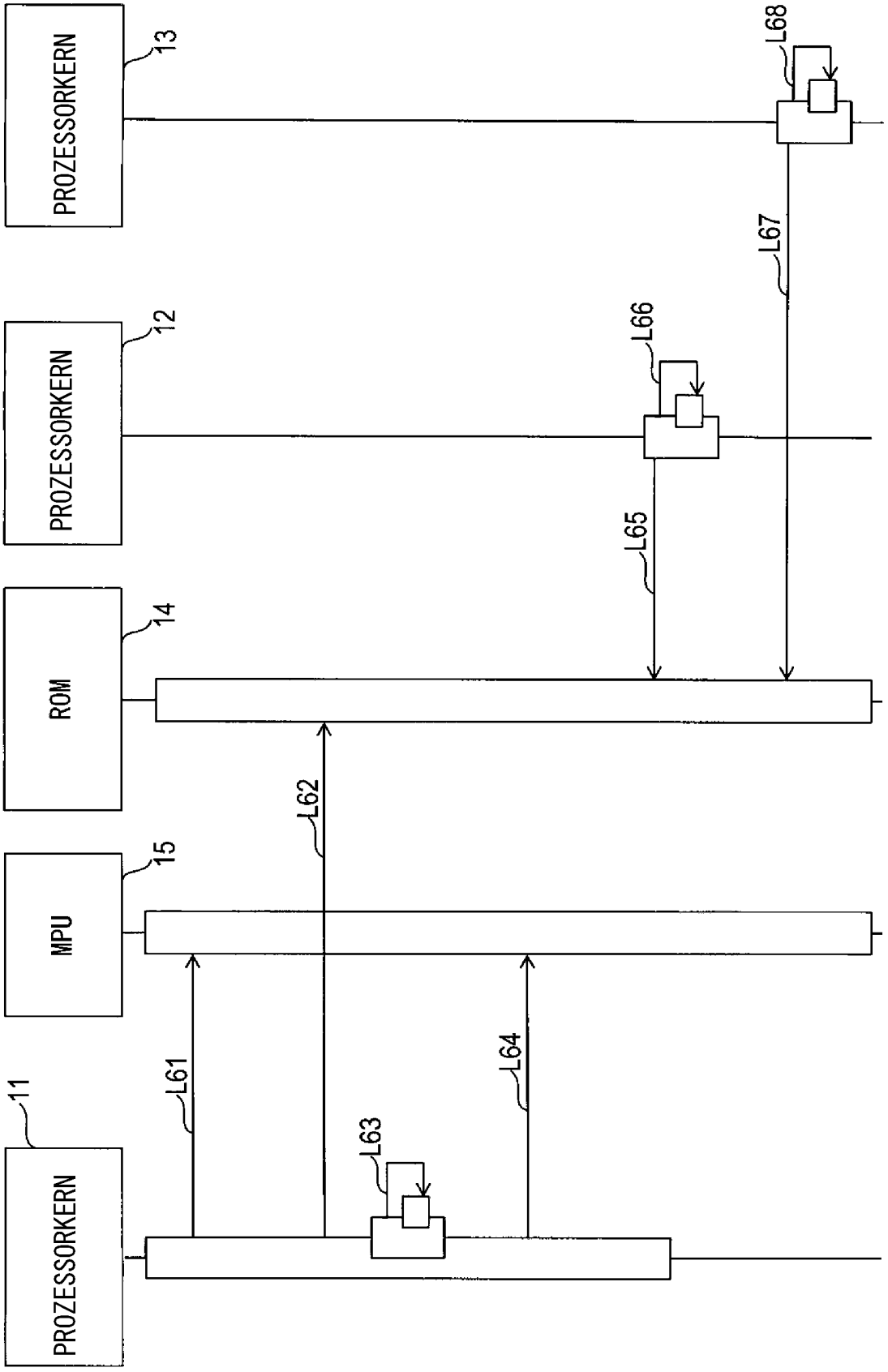

PROZESSORKERN
13
L68
L67
PROZESSORKERN
12
L66
L65
ROM
14
L62
MPU
15
L61
L64
11
L63
PROZESSORKERN

FIG. 16

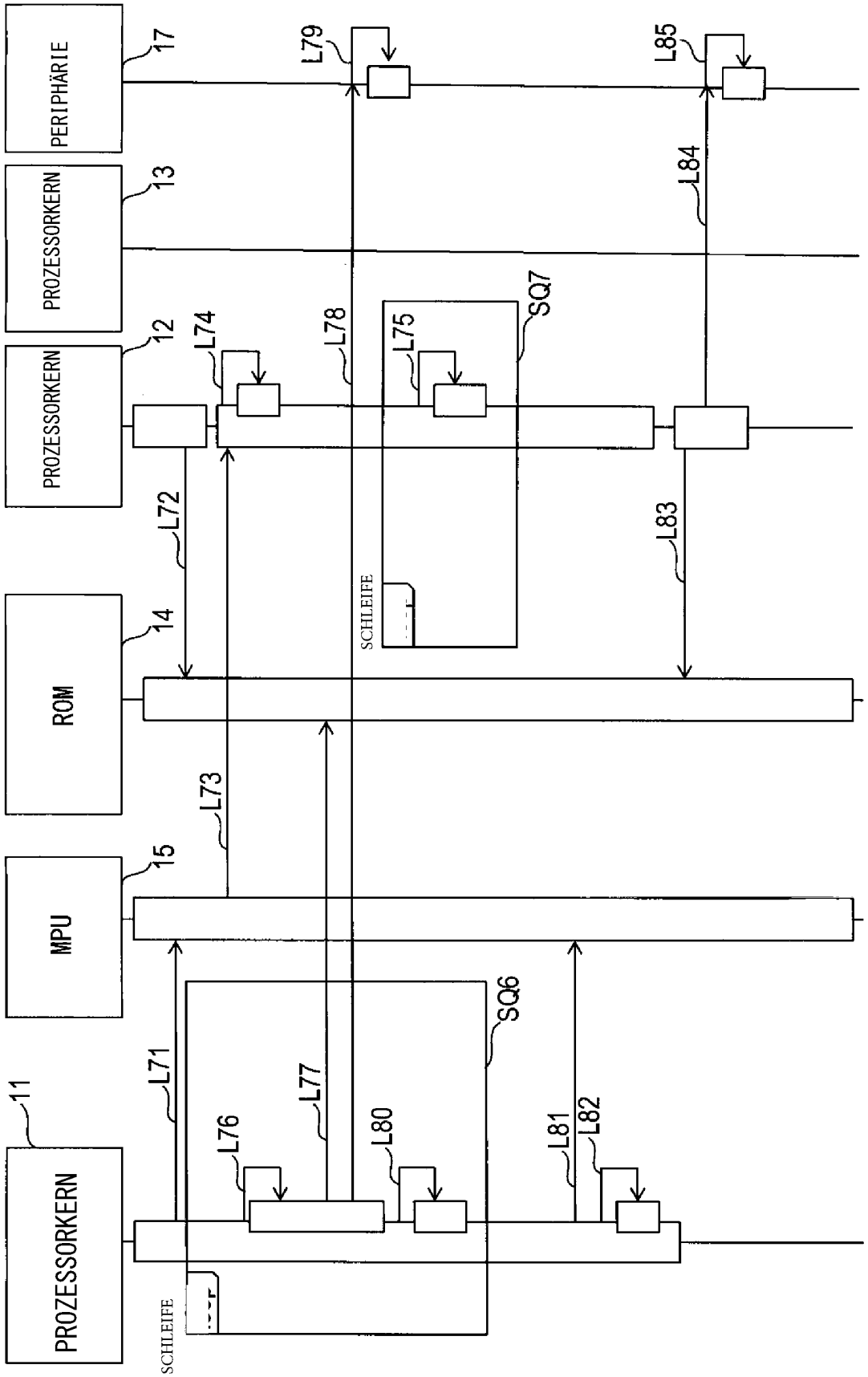

PERIPHÄRIE
17
PROZESSORKERN
13
PROZESSORKERN
12
ROM
14
MPU
15
PROZESSORKERN
11
L71
L72
L73
L74
L75
L76
L77
L78
L79
L80
L81
L82
L83
L84
L85
SQ6
SQ7
SCHLEIFE
SCHLEIFE

FIG. 17