

Beschreibung

Verfahren zum Erzeugen und Abspeichern eines aus Befehlen bestehenden Anwenderprogramms für eine speicherprogrammierbare Steuerung und Betriebsverfahren für eine speicherprogrammierbare Steuerung

Die vorliegende Erfindung betrifft ein Verfahren zum Erzeugen und Abspeichern eines aus Befehlen bestehenden Anwenderprogramms für eine speicherprogrammierbare Steuerung, wobei die speicherprogrammierbare Steuerung bei Abarbeitung des Anwenderprogramms aus Eingangssignalen von einem technischen Prozeß Ausgangssignale für den technischen Prozeß ermittelt, sowie ein hierzu korrespondierendes Betriebsverfahren für eine speicherprogrammierbare Steuerung.

Derartige Verfahren sind allgemein bekannt.

Die Aufgabe der vorliegenden Erfindung besteht darin, neue Verfahren zur Verfügung zu stellen.

Die Aufgabe wird für das Erzeugungs- und Abspeicherverfahren dadurch gelöst, daß die Befehle des Anwenderprogramms jeweils aus einem Funktionscode und dem Funktionscode zugeordneten Funktionsparametern bestehenden, daß die Funktionsparameter entweder auf ein Eingangssignal oder auf einen anderen Befehl verweisen und daß das Verfahren folgende Schritte aufweist:

- a) bei der Erzeugung des Anwenderprogramms werden pro Ausgangssignal, beginnend mit dem dem jeweiligen Ausgangssignal zuzuordnenden Befehl, die abzuarbeitenden Befehle abgefragt, wobei folgende Prozedur abgearbeitet wird:
- b) es wird der abzuarbeitende Funktionscode abgefragt und zwischengespeichert;
- c) es wird die Zahl der Funktionsparameter festgestellt;
- d) es werden nacheinander die Funktionsparameter abgefragt;

2

- e) wenn der Funktionsparameter ein Eingangssignal ist, wird ein Verweis auf das Eingangssignal zwischengespeichert;
- f) wenn der Funktionsparameter ein neuer Funktionscode ist, wird ein Verweis auf einen neuen Befehl zwischengespeichert, wird zu diesem neuen Befehl verzweigt und werden der andere Funktionscode zwischengespeichert und die Schritte c) bis f) für den neuen Befehl wiederholt;
- g) die zwischengespeicherten Funktionscodes und die zwischengespeicherten Funktionsparameter werden erst dann in einem Remanentspeicher hinterlegt bzw. als gültig gekennzeichnet, wenn alle Befehle zumindest des jeweiligen Ausgangssignals vollständig abgefragt sind.

Auch gemäß dem Betriebsverfahren bestehen die Befehle des Anwenderprogramms jeweils aus einem Funktionscode und dem Funktionscode zugeordneten Funktionsparametern, wobei die Funktionsparameter entweder auf ein Eingangssignal oder auf einen anderen Befehl verweisen. Die erfindungsgemäßen Schritte sind wie folgt:

- a) bei der zyklischen Abarbeitung wird, beginnend mit den den Ausgangssignalen zugeordneten Befehlen, pro Befehl folgende Prozedur abgearbeitet;
- b) es werden der dem Befehl zugeordnete Funktionscode und die diesem Befehl zugeordneten Funktionsparameter gelesen;
- c) pro Funktionsparameter wird überprüft, ob dieser auf einen anderen Befehl verweist;
- d) wenn der Funktionsparameter auf ein Eingangssignal verweist, wird das Eingangssignal zwischengespeichert;
- e) wenn der Funktionsparameter auf einen anderen Befehl verweist, wird zu dem anderen Befehl verzweigt und werden die Schritte b) bis e) mit dem anderen Befehl wiederholt;
- f) wenn alle Funktionsparameter abgearbeitet sind, wird das dem Befehl zugeordnete Ausgangssignal ermittelt.

Weitere Vorteile und Einzelheiten ergeben sich aus der nachfolgenden Beschreibung eines Ausführungsbeispiels. Dabei zeigen:

- 5 FIG 1 den Aufbau eines Programmiergeräts und einer speicherprogrammierbaren Steuerung,
FIG 2 die Benutzerschnittstelle des Programmiergeräts und
FIG 3 ein Programmierbeispiel.
- 10 Gemäß FIG 1 weist das Programmiergerät 1 einen Prozessor 2 auf, der über das Bussystem 3 mit dem Festwertspeicher 4 und dem Schreib-Lese-Speicher 5 sowie den Schnittstellen 6,7 und 8 verbunden ist. Im Festwertspeicher 4 ist das Betriebssystem des Prozessors 2 abgespeichert, der Schreib-Lese-Speicher 5
15 dient dem Programmiergerät 1 als Arbeitsspeicher. Über die Schnittstelle 6 sind mittels der Eingabeeinheit 9, z.B. einer Tastatur, Daten in das Programmiergerät 1 eingebbar. Ebenso sind mittels der Schnittstelle 7 Daten auf der Anzeigeeinheit
20 10 darstellbar. Mittels der Kommunikationsschnittstelle 8 kann das Programmiergerät über die gestrichelt eingezeichnete Kommunikationsverbindung 11 mit einer speicherprogrammierbaren Steuerung 12 kommunizieren.
- Die speicherprogrammierbare Steuerung 12 weist ebenfalls ei-
25 nen Prozessor 13 auf, der ein Anwenderprogramm abarbeitet. Das Anwenderprogramm besteht aus einer Vielzahl von auszuführenden Befehlen und ist im elektrisch lösch- und schreibbaren Festwertspeicher 14 hinterlegt. Gemäß dem abzuarbeitenden Anwenderprogramm greift der Prozessor 13 über den Controller 15
30 und die Prozeßschnittstelle 16 auf einen nicht näher dargestellten technischen Prozeß bzw. eine technische Anlage 17 zu. Er liest von dort Eingangssignale ein, speichert sie im Schreib-Lese-Speicher 18 ab und ermittelt sodann anhand des Anwenderprogramms Ausgangssignale. Die Ausgangssignale werden
35 ebenfalls im Schreib-Lese-Speicher 18 hinterlegt und dann über den Controller 15 und die Prozeßschnittstelle 16 an den

technischen Prozeß 17 ausgegeben. Ferner weist die speicherprogrammierbare Steuerung 12 eine Schnittstelle 19 zum Programmiergerät auf. Der Prozessor 13, die Speicher 14,18, der Controller 15 sowie die Kommunikationsschnittstelle 19 sind
5 ebenfalls über einen Bus 20 miteinander verbunden.

Programmiergerät 1 und speicherprogrammierbare Steuerung 12 können im Prinzip voneinander getrennte Geräte sein. Vorzugsweise ist das Programmiergerät 1 jedoch, wie in FIG 1 durch
10 die gestrichelte Linie und auch in FIG 2 dargestellt, in die speicherprogrammierbare Steuerung 12 integriert. In beiden Fällen ist es aber möglich, daß das Programmiergerät 1 über die Schnittstellen 8 und 19 sowie die Kommunikationsverbindung 11 auf den Festwertspeicher 14 zugreift. Dabei ist es
15 möglich, daß ein neues Anwenderprogramm in den Festwertspeicher 14 geschrieben wird und daß ein bestehendes Anwenderprogramm gelesen, gelöscht oder geändert wird.

Gemäß FIG 2 weist die speicherprogrammierbare Steuerung 12
20 Schraubanschlüsse 21 bis 24, ein Tastaturfeld 25 sowie ein LCD-Display 26 auf. Über die Schraubanschlüsse 21 ist die speicherprogrammierbare Steuerung 12 mit einer 24 Volt-Stromversorgung verbindbar. Über je zwei der Schraubanschlüsse 22 ist ein Digital-Ausgangssignal an den technischen Prozeß 17
25 ausgebar. Über je eine der Schraubklemmen 23 ist ein Eingangssignal vom technischen Prozeß 17 einlesbar. Über die Schraubklemmen 24 sind die Aktoren und Sensoren, welche an die Schraubklemmen 22 und 23 angeschlossen sind, mit Strom versorgbar. Die Schraubklemmen 22 bis 24 sind also Bestand-
30 teil der Prozeßschnittstelle 16.

Das LCD-Display 26 entspricht der Anzeigeeinheit 10 des Programmiergeräts 1. Über das Display 26 sind ferner auch Meldungen der speicherprogrammierbaren Steuerung 12 während des Betriebs anzeigbar. Das Display 26 dient also auch als Beob-
35 achtungseinheit. Das Tastaturfeld 25 ermöglicht es, Befehle

direkt in das Programmiergerät 1 und die speicherprogrammierbare Steuerung 12 einzugeben.

Das Anwenderprogramm besteht, wie allgemein üblich aus einzelnen Befehlen, wobei jeder Befehl jeweils aus einem Funktionscode und dem Funktionscode zugeordneten Funktionsparametern besteht. Im Betrieb arbeitet die speicherprogrammierbare Steuerung zyklisch folgende Befehlsfolge ab:

- 10 - Sie liest Eingangssignale vom technischen Prozeß 17 ein.
- Sie ermittelt anhand des Anwenderprogramms Ausgangssignale für den technischen Prozeß.
- Sie gibt die Ausgangssignale an den technischen Prozeß 17 aus.

15

Diese Vorgehensweise ist allgemein bekannt.

Bei der Erzeugung des Anwenderprogramms werden durch die Programmiereinheit 1 nacheinander die abzuarbeitenden Befehle von der Eingabeeinheit 9 abgefragt. Hierzu wird, beginnend mit dem dem ersten Ausgangssignal zuzuordnenden Befehl, abgefragt, welche Funktion abgearbeitet werden soll. Es wird also eine entsprechende Anfrage auf der Anzeigeeinheit 10 ausgegeben und gewartet, bis der Anwender über die Eingabeeinheit 9 den Funktionscode eingegeben hat.

Sowie der Funktionscode eingegeben ist, wird dieser von der Programmiereinheit 1 im Schreib-Lese-Speicher 5 zwischengespeichert. Sodann stellt die Programmiereinheit 1 fest, wie viele Funktionsparameter diesem Befehl zugeordnet sind. Hierbei ist zu unterscheiden zwischen der internen Zuordnung innerhalb des Programmiergeräts 1 und der für den Anwender sichtbaren Zahl der Funktionsparameter. Intern ist die Zahl der Funktionsparameter für alle Funktionscodes derselbe, im vorgegebenen Beispiel gleich vier. Nach außen hingegen wird

stets nur die Zahl der benötigten Funktionsparameter abgefragt. Ein Beispiel soll dies näher erläutern:

Wenn als Funktionscode der Code für die Oderverknüpfung eingegeben wird, werden tatsächlich vier Parameter abgefragt. Diese vier Parameter werden dann später bei der Abarbeitung des Programms auch berücksichtigt. Wenn hingegen z.B. die logische Verneinung als Funktionscode gewählt wird, wird vom Anwender nur ein Parameter, nämlich das zu negierende Eingangssignal, abgefragt. Die anderen drei Parameter werden vom Programmiergerät 1 selbständig aufgefüllt und bei der Abarbeitung des Programms nicht berücksichtigt.

Nach dem Abfragen und Zwischenspeichern des Funktionscodes im Schreib-Lese-Speicher 5 werden durch die Programmereinheit 1 im Zusammenwirken mit der Eingabeeinheit 9 und deranzeigeeinheit 10 nacheinander auch die Funktionsparameter abgefragt und zwischengespeichert. Als Funktionsparameter kann entweder ein Eingangssignal oder ein anderer Funktionscode eingegeben werden. Dabei wird bei jeder Eingabe eines Funktionsparameters sofort überprüft, ob dieser ein Funktionscode ist.

Wenn der Funktionsparameter kein Funktionscode, sondern ein Eingangssignal ist, wird dieses zwischengespeichert und mit dem nächsten Funktionsparameter fortgefahren. Unter Eingangssignal sind dabei im vorliegenden Fall nicht nur Eingangssignale vom technischen Prozeß 17 zu verstehen, sondern auch Signale, deren Wert vorbekannt ist. Beispiel für derartige Signale sind das Konstant-Eins- oder das Konstant-Null-Signal sowie ein Taktsignal.

Wenn als Funktionsparameter dagegen ein neuer Funktionscode eingegeben wird, wird als Funktionsparameter nicht der Funktionscode zwischengespeichert, sondern die Adresse eines neuen Befehls. Unmittelbar danach wird zu diesem neuen Befehl verzweigt und dort der unmittelbar zuvor eingegebene Funk-

tionscode zwischengespeichert. Sodann wird erneut die Zahl der Funktionsparameter festgestellt, und es werden die Funktionsparameter abgefragt und zwischengespeichert.

- 5 Auch bei diesen Funktionsparametern wird sofort wieder überprüft, ob sie Eingangssignale oder Funktionscodes sind. Wenn sie neue Funktionscodes sind, wird wiederum erneut sofort die Adresse dieses neuen Befehls als Funktionsparameter abgespeichert und zu diesem neuen Befehl verzweigt. Auch hier wird
10 wieder der soeben eingegebene Funktionscode abgespeichert, festgestellt wieviele Funktionsparameter benötigt werden und die Funktionsparameter abgefragt und zwischengespeichert.

- Dieses verschachtelte Abfragen von Funktionscodes und Funktionsparametern wird solange fortgesetzt, bis alle Funktionsparameter des zuletzt abgefragten und zwischengespeicherten Funktionscodes auf Eingangssignale verweisen. Wenn dann alle Funktionsparameter auf Eingangssignale verweisen, ist der jeweilige Befehl vollständig eingegeben. Es wird dann zurückgegangen zum vorhergehenden Befehl und bei diesem gegebenenfalls der nächste Funktionsparameter abgefragt und zwischengespeichert, wobei auch hier wieder gegebenenfalls zu weiteren Befehlen verzweigt wird.
15
20

- 25 Obenstehende Prozedur wird solange fortgesetzt, bis alle zur Ermittlung eines Ausgangssignals erforderlichen Funktionen miteinander verschaltet sind, also alle Funktionsparameter jedes Befehls eingegeben worden sind. Erst wenn dies erfolgt ist, wird der dem nächsten Ausgangssignal zuzuordnende Befehl
30 abgefragt und die Prozedur, soweit erforderlich, wiederholt. Auf diese Art werden nacheinander alle Ausgangssignale programmiert.

- Das Eingeben des Anwenderprogramms kann vom Anwender selbstverständlich jederzeit abgebrochen werden. In diesem Fall erfolgt jedoch kein Abspeichern des eingegebenen und zwischen-
35

- gespeicherten Teilprogramms. Beim Ausschalten des Programmiergeräts 1 sind die eingegebenen Daten verloren. Ein Abspeichern der abgefragten und im Schreib-Lese-Speicher 5 zwischengespeicherten Anwenderprogramms im Remanentspeicher 14
- 5 wird erst dann ermöglicht, wenn der Programmiervorgang vollständig beendet ist. Als vollständig beendet gilt der Programmiervorgang bzw. das Erzeugen des Anwenderprogramms dann,
- wenn zu jedem Ausgangssignal ein Befehl eingegeben worden ist, und
- 10 - wenn jeder Funktionsparameter jedes programmierten Funktionscodes belegt ist.

FIG 3 zeigt ein Beispiel eines einfachen Anwenderprogramms. Die durchgezogenen Linien bedeuten dabei die aufgrund der gewünschten Verknüpfungen erforderlichen Verbindungen. Die gestrichelten Linien zeigen die aufgrund des Programmierschemas erforderlichen zusätzlichen Verbindungen, und die strichpunkt

15 tierten Linien zeigen die vom Programmiergerät 1 selbständig ergänzten Verbindungen. Die Eingabe des Programmierbeispiels gemäß FIG 3 geschieht wie folgt:

20

Zunächst fragt das Programmiergerät über Anzeigeeinheit 10 und Eingabeeinheit 9 ab, welcher Funktionscode für die Ermittlung des Ausgangssignals des ersten Ausgangs A1 benötigt

25 wird. Der Anwender gibt daraufhin über die Eingabeeinheit ein, daß eine UND-Verknüpfung 27 erfolgen soll. Der korrespondierende Funktionscode wird im Schreib-Lese-Speicher 5 des Programmiergeräts 1 zwischengespeichert. Sodann werden nacheinander der erste, der zweite und der dritte Funktionsparameter, d.h. drei der vier Eingangssignale der UND-Verknüpfung

30 27, abgefragt und zwischengespeichert.

Der erste und der zweite Eingangsparameter sind die Eingangssignale E1 und E2. Es wird daher bei der Eingabe dieser beiden Funktionsparameter sofort mit dem nächsten Funktionsparameter fortgefahren. Beim Abfragen des dritten Funktionspa

35

rameters hingegen wird durch den Anwender eingegeben, daß er eine NICHT-Verknüpfung 28 vorzunehmen wünscht. Als dritter Funktionsparameter wird daher die Adresse des Befehls zwischengespeichert, an dem dann die NICHT-Verknüpfung 28 behandelt wird.

Sodann wird durch das Programmiergerät 1 zu diesem Befehl verzweigt und abgefragt, welches der (einzige) Funktionsparameter der NICHT-Verknüpfung 28 sein soll. Hier wird durch den Anwender das Eingangssignal E3 eingegeben. Das Programmiergerät 1 speichert diesen Funktionsparameter zwischen und ergänzt den Befehl mit drei weiteren Funktionsparametern, welche jeweils auf die logische Eins verweisen. Auch diese drei Funktionsparameter werden im Speicher 5 zwischengespeichert. Hierdurch ist dann die Parametrierung der NICHT-Verknüpfung 28 abgeschlossen. Es wird daher zur weiteren Parametrierung der UND-Verknüpfung 27 zurückgekehrt und der vierte Funktionsparameter abgefragt und zwischengespeichert. Da die UND-Verknüpfung 27 gemäß dem Beispiel nach FIG 3 nur drei echte Eingangsparameter aufweist, wird der vierte Eingang durch entsprechende Eingabe des Anwenders mit der logischen Eins verbunden.

Hiermit ist die Programmierung des Ausgangs A1 abgeschlossen, und es wird mit der Programmierung des Ausgangs A2 fortgeföhren.

Zunächst wird abgefragt und nach Eingabe durch den Anwender zwischengespeichert, daß der Funktionscode für die Erzeugung des Ausgangssignals A2 eine ODER-Verknüpfung 29 ist. Sodann werden nacheinander die vier Funktionsparameter der ODER-Verknüpfung 29 abgefragt.

Als ersten Funktionsparameter wird gemäß dem Programmierbeispiel von FIG 3 die bereits programmierte NICHT-Verknüpfung des Eingangs E3 eingegeben. Da diese Verknüpfung bereits fer-

10

tig programmiert ist, ist keine weitere Eingabe bezüglich dieses Funktionsparameters erforderlich und es wird nach der Zwischenspeicherung sofort mit dem zweiten Funktionsparameter fortgefahren.

5

Der zweite und der dritte Funktionsparameter sind die (echten) Eingangssignale E2 und E4, so daß sofort mit dem vierten Funktionsparameter fortgefahren werden kann.

- 10 Gemäß dem Programmierbeispiel von FIG 3 hat auch die ODER-Verknüpfung 29 nur drei Eingangssignale. Der vierte Funktionsparameter verweist daher auf die Konstant-Null.

- Da hiermit auch die Programmierung des Ausgangs A2 abgeschlossen ist, wird mit dem Ausgang A3 fortgefahren.
- 15

- Die Ausgänge A3 und A4 werden gemäß dem gegebenen Programmierbeispiel eigentlich nicht benötigt. Für den Ausgang A3 wird daher die NICHT-Verknüpfung 30 gewählt, welche mit der logischen Eins verknüpft wird. Dadurch ist der Ausgang A3 auf Konstant-Null geschaltet. Gleiches geschieht mit dem Ausgang A4, für welchen die NICHT-Verknüpfung 31 eingegeben wird. Alternativ ist es auch möglich, die Ausgänge A3 und A4 direkt mit der logischen Null zu verbinden.
- 20

25

Das daraus resultierende Programm wird dann im Speicher 14 abgespeichert. Es ist in der nachstehenden Tabelle dargestellt.

- 30 B1: FC (AND)
ADDR (E1)
ADDR (E2)
ADDR (B5)
ADDR ("1")
- 35 B2: FC (OR)
ADDR (B5)

11

```

        ADDR (E2)
        ADDR (E4)
        ADDR ("0")
B3:  FC  (NOT)
5    ADDR ("1")
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")
B4:  FC  (NOT)
10   ADDR ("1")
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")
B5:  FC  (NOT)
15   ADDR (E3)
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")

```

20 Links in Tabelle 1 sind mit B1 bis B5 jeweils der Anfang der einzelnen Befehle dargestellt. Diese Angaben dienen aber nur zur Erläuterung und sind nicht Bestandteil des abgespeicherten Programms. Im eigentlichen Programm steht FC für Funktionscode, ADDR für Adresse.

25

Das Programm als solches besteht aus Blöcken zu je fünf Byte. Das erste Byte stellt jeweils den Funktionscode des Befehls dar, die anderen vier Bytes die vier Funktionsparameter. Die ersten vier Befehle B1 bis B4 definieren stets die Befehle für die vier Ausgangssignale. Ab dem fünften Befehl B5 sind dann Befehle abgespeichert, auf die bei der Berechnung der Ausgangssignale indirekt zurückgegriffen werden muß.

30

Die Abarbeitung des Anwenderprogramms geschieht wie folgt:

35

- Es werden zyklisch die Befehle B1 bis B4 abgearbeitet. Bei der Abarbeitung jedes der Befehle B1 bis B4 wird zunächst der dem Befehl zugeordnete Funktionscode gelesen. Sodann werden nacheinander die diesem Befehl zugeordneten Funktionsparameter
- 5 gelesen. Beim Lesen jedes Funktionsparameters wird sofort überprüft, ob dieser auf einen anderen Befehl verweist. Wenn der Funktionsparameter auf ein Eingangssignal verweist, wird dieses zwischengespeichert.
- 10 Wenn der Funktionsparameter dagegen auf einen anderen Befehl verweist, wird sofort, d.h. vor dem Lesen der anderen Funktionsparameter, zu diesem Befehl verzweigt und der diesen Befehl zugeordnete Funktionscode und die diesem Befehl zugeordneten Funktionsparameter gelesen. Auch hier wird, wenn der
- 15 Funktionscode auf einen neuen Befehl zeigt, sofort wieder dieser Befehl ausgeführt, bevor die weiteren Funktionsparameter abgearbeitet werden. Wenn alle Funktionsparameter eines Befehls abgearbeitet sind, wird das diesem Befehl zugeordnete Ausgangssignal ermittelt, gegebenenfalls zum aufrufenden Befehl
- 20 zurückgesprungen, und das ermittelte Ausgangssignal der weiteren Abarbeitung des Anwenderprogramms zugrunde gelegt.

Auch hier soll das Prinzip anhand des Programms gemäß Tabelle näher erläutert werden.

- 25
- Zunächst wird also der Funktionscode des Befehls B1 gelesen. Sodann werden die Funktionsparameter gelesen, welche auf die Eingänge E1 und E2 verweisen. Beim Lesen des dritten Funktionsparameters wird zum Befehl B5 verzweigt. Dessen Funktionscode und alle vier Funktionsparameter (weil alle auf
- 30 Eingangssignale verweisen) werden gelesen und das dem Befehl B5 zugeordnete Ausgangssignal ermittelt. Mit diesem Ausgangssignal wird dann zum Befehl B1 zurückgekehrt und der vierte Funktionsparameter gelesen. Sodann wird das Ausgangssignal A1
- 35 ermittelt und im Schreib-Lese-Speicher 18 abgespeichert.

Danach wird der Funktionscode des Befehls B2 gelesen und der erste Funktionsparameter des Befehls B2. Sodann wird erneut zum Befehl B5 verzweigt, dessen Funktionscode und dessen vier Funktionsparameter gelesen und das Ausgangssignal des Befehls B5 ermittelt. Sodann wird zum Befehl B2 zurückgekehrt und die übrigen drei Funktionsparameter gelesen. Sodann wird das Ausgangssignal des Befehls B2 ermittelt.

Schließlich werden noch der Funktionscode und die Funktionsparameter des Befehls B3 gelesen und ausgeführt und so das Ausgangssignal A3 ermittelt. Gleiches geschieht mit dem Befehl B4 und dem daraus folgenden Ausgangssignal A4.

Nach der Ermittlung des letzten Ausgangssignals A4 werden die so ermittelten Ausgangssignale A1 bis A4 an den technischen Prozeß 17 ausgegeben und erneut die Eingangssignale E1 bis E4 eingelesen. Sodann wird ein neuer Zyklus bei der Ermittlung der Ausgangssignale A1 bis A4 begonnen.

Alternativ zum beschriebenen Erstellungsverfahren ist es auch möglich, die Befehle, die zu einem Ausgangssignal gehören, sofort im Remanentspeicher 14 abzuspeichern, wenn das jeweilige Ausgangssignal vollständig programmiert ist. Es könnten beispielsweise also die Befehle B1 und B5 sofort abgespeichert werden, sowie das Ausgangssignal A1 vollständig programmiert ist.

Ebenso ist es auch möglich, jeden eingegebenen Funktionscode und jeden eingegebenen Funktionsparameter sofort im Remanentspeicher 14 abzuspeichern. In diesem Fall müßte in einem speziell hierfür vorgesehenen Speicherbereich des Remanentspeichers 14 zu Beginn eines Programmerstellungsvorgangs hinterlegt werden, daß die Programmierung aller Ausgangssignale A1 bis A4 ungültig ist. Beispielsweise könnten vier 1 Bit-Speicherzellen auf logisch-Null gesetzt werden, um anzudeuten, daß die vier Ausgangssignale A1 bis A4 ungültig programmiert

sind. Die vier 1 Bit-Speicherzellen könnten dann auf logisch-Eins gesetzt werden, wenn die Ausgangssignale A1 bis A4 gültig programmiert worden sind. Das Als-gültig-Kennzeichnen kann dabei wahlweise nach der Erstellung des gesamten Anwenderprogramms oder separat für jedes Ausgangssignal A1 bis A4 nach der Abfrage aller Befehle des jeweiligen Ausgangssignals A1 bis A4 erfolgen.

Die Befehle müssen auch nicht in der Reihenfolge gemäß der
10 Tabelle abgespeichert sein. Es ist beispielsweise möglich, alle Befehle, die zur Ermittlung des Ausgangssignals A1 benötigt werden, hier die Befehle B1 und B5, nacheinander im Speicher 14 abzulegen. Die Befehle B2 bis B4 zur Ermittlung der Ausgangssignale A2 bis A4 würden dann hinter diesen Befehlen (hier B1 und B5) abgespeichert werden. Beim Abarbeiten
15 des Anwenderprogramms ist in diesem Fall zu verfolgen, welches die höchste Adresse ist, die zur Ermittlung des Ausgangssignals A1 aufgerufen wird. Der Befehl zur Ermittlung des Ausgangssignals A2 beginnt dann bei der nächsthöheren
20 Adresse. Auf ähnliche Weise, nämlich Zwischenspeichern der höchsten benötigten Adresse, kann dann ermittelt werden, an welcher Stelle die Befehle zur Ermittlung der Ausgangssignale A3 und A4 angeordnet sind. Ob der Fachmann bei der Implementierung der vorliegenden Erfindung der einen oder der anderen
25 Vorgehensweise den Vorzug gibt, liegt jedoch in seinem Belieben.

Patentansprüche

1. Verfahren zum Erzeugen und Abspeichern eines aus Befehlen (B1 bis B5) bestehenden Anwenderprogramms für eine speicher-
5 programmierbare Steuerung,
- wobei die speicherprogrammierbare Steuerung bei Abarbeitung des Anwenderprogramms aus Eingangssignalen (E1 bis E4) von einem technischen Prozeß (17) Ausgangssignale (A1 bis A4) für den technischen Prozeß (17) ermittelt,
10 - wobei die Befehle (B1 bis B5) des Anwenderprogramms jeweils aus einem Funktionscode und dem Funktionscode zugeordneten Funktionsparametern bestehen,
- wobei die Funktionsparameter entweder auf ein Eingangssignal (E1 bis E4) oder auf einen anderen Befehl (B5) verwei-
15 sen,
mit folgenden Schritten:
a) bei der Erzeugung des Anwenderprogramms werden pro Ausgangssignal, beginnend mit dem dem jeweiligen Ausgangssignal (A1 bis A4) zuzuordnenden Befehl (B1 bis B4), die
20 abzuarbeitenden Befehle (B1 bis B5) abgefragt, wobei folgende Prozedur abgearbeitet wird:
b) es wird der abzuarbeitende Funktionscode abgefragt und zwischengespeichert;
c) es wird die Zahl der Funktionsparameter festgestellt;
25 d) es werden nacheinander die Funktionsparameter abgefragt;
e) wenn der Funktionsparameter ein Eingangssignal (E1 bis E4) ist, wird ein Verweis auf das Eingangssignal (E1 bis E4) zwischengespeichert;
f) wenn der Funktionsparameter ein neuer Funktionscode ist,
30 wird ein Verweis auf einen neuen Befehl (B5) zwischengespeichert, wird zu diesem neuen Befehl (B5) verzweigt und werden der andere Funktionscode zwischengespeichert und die Schritte c) bis f) für den neuen Befehl (B5) wiederholt;
35 g) die zwischengespeicherten Funktionscodes und die zwischengespeicherten Funktionsparameter werden erst dann in einem



Remanentspeicher (14) hinterlegt bzw. erst dann als gültig gekennzeichnet, wenn alle Befehle (B1 bis B5) zumindest des jeweiligen Ausgangssignals (A1 bis A4) vollständig abgefragt sind.

5

2. Betriebsverfahren für eine speicherprogrammierbare Steuerung,

- wobei die speicherprogrammierbare Steuerung zyklisch ein aus Befehlen (B1 bis B5) bestehendes Anwenderprogramm abarbeitet, um aus Eingangssignalen (E1 bis E4) von einem technischen Prozeß (17) Ausgangssignale (A1 bis A4) für den technischen Prozeß (17) zu ermitteln,
- wobei die Befehle (B1 bis B5) des Anwenderprogramms jeweils aus einem Funktionscode und dem Funktionscode zugeordneten Funktionsparametern bestehen,
- wobei die Funktionsparameter entweder auf ein Eingangssignal (E1 bis E4) oder auf einen anderen Befehl (B5) verweisen,

mit folgenden Schritten:

- a) bei der zyklischen Abarbeitung wird, beginnend mit den den Ausgangssignalen (A1 bis A4) zugeordneten Befehlen (B1 bis B4), pro Befehl (B1 bis B5) folgende Prozedur abgearbeitet:
- b) es werden der dem Befehl (B1 bis B5) zugeordnete Funktionscode und die diesem Befehl (B1 bis B5) zugeordneten Funktionsparameter gelesen;
- c) pro Funktionsparameter wird überprüft, ob dieser auf einen anderen Befehl (B5) verweist;
- d) wenn der Funktionsparameter auf ein Eingangssignal (E1 bis E4) verweist, wird das Eingangssignal (E1 bis E4) zwischengespeichert;
- e) wenn der Funktionsparameter auf einen anderen Befehl (B5) verweist, wird zu dem anderen Befehl (B5) verzweigt und werden die Schritte b) bis e) mit dem anderen Befehl (B5) wiederholt;



- f) wenn alle Funktionsparameter abgearbeitet sind, wird das dem Befehl (B1 bis B5) zugeordnete Ausgangssignal (A1 bis A4) ermittelt.
- 5 3. Programmiereinheit zur Durchführung des Verfahren nach Anspruch 1, mit
- Mitteln (2,4,5,6,7,9,10) zum Abfragen und Zwischenspeichern von Funktionscodes abzuarbeitender Befehle;
 - Mitteln (2,4) zum Feststellen der Zahl von Funktionsparametern der abzuarbeitenden Befehle;
 - 10 - Mitteln (2,4,5,6,7,9,10) zum Abfragen und Zwischenspeichern der Funktionsparameter;
 - Mitteln (2,4) zum Überprüfen der Funktionsparameter, ob diese ein neuer Funktionscode sind;
 - 15 - Mitteln (2,4) zum Verzweigen zu einem neuen Befehl, wenn der Funktionsparameter ein neuer Funktionscode ist;
 - Mitteln (2,4) zum Verhindern des Abspeicherns der zwischengespeicherten Funktionscodes und der zwischengespeicherten Funktionsparameter in einem Remanentspeicher (14) bzw. zum
 - 20 Verhindern des Als-gültig-Kennzeichnens, wenn noch nicht alle Funktionsparameter aller abzuarbeitenden Befehle zumindest eines Ausgangssignals (A1 bis A4) zwischengespeichert sind.
- 25 4. Speicherprogrammierbare Steuerung zur Durchführung des Verfahren nach Anspruch 2, mit
- Mitteln (13) zum Lesen von Befehlen zugeordneten Funktionscodes und Funktionsparametern;
 - Mitteln (13) zum Überprüfen der Funktionsparameter, ob diese auf ein Eingangssignal oder auf einen anderen Befehl
 - 30 verweisen;
 - Mitteln (18) zum Zwischenspeichern des Eingangssignals, wenn der Funktionsparameter auf das Eingangssignal verweist;
 - 35 - Mitteln (13) zum Verzweigen zu dem anderen Befehl, wenn der Funktionsparameter auf den anderen Befehl verweist;



18

- Mitteln (13) zum Zurückkehren zu einem aufrufenden Befehl;
- Mitteln (13) zum Ermitteln des dem Befehl zugeordneten Ausgangssignals, wenn alle ihm zugeordneten Funktionsparameter abgearbeitet sind.

5

5. Speicherprogrammierbare Steuerung nach Anspruch 4, die mit einer Programmiereinheit nach Anspruch 3 kombiniert ist.



1/3

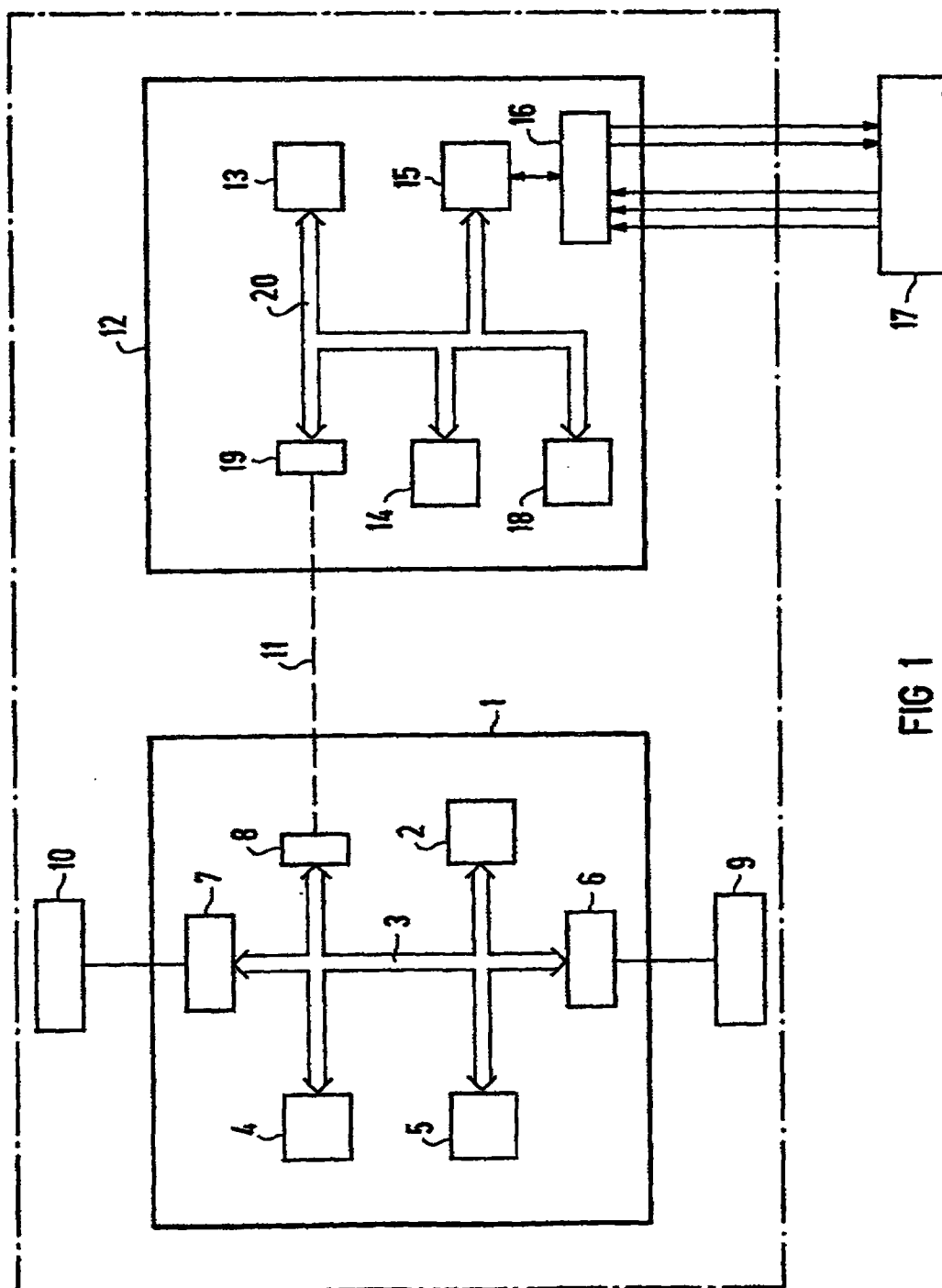


FIG 1

2/3

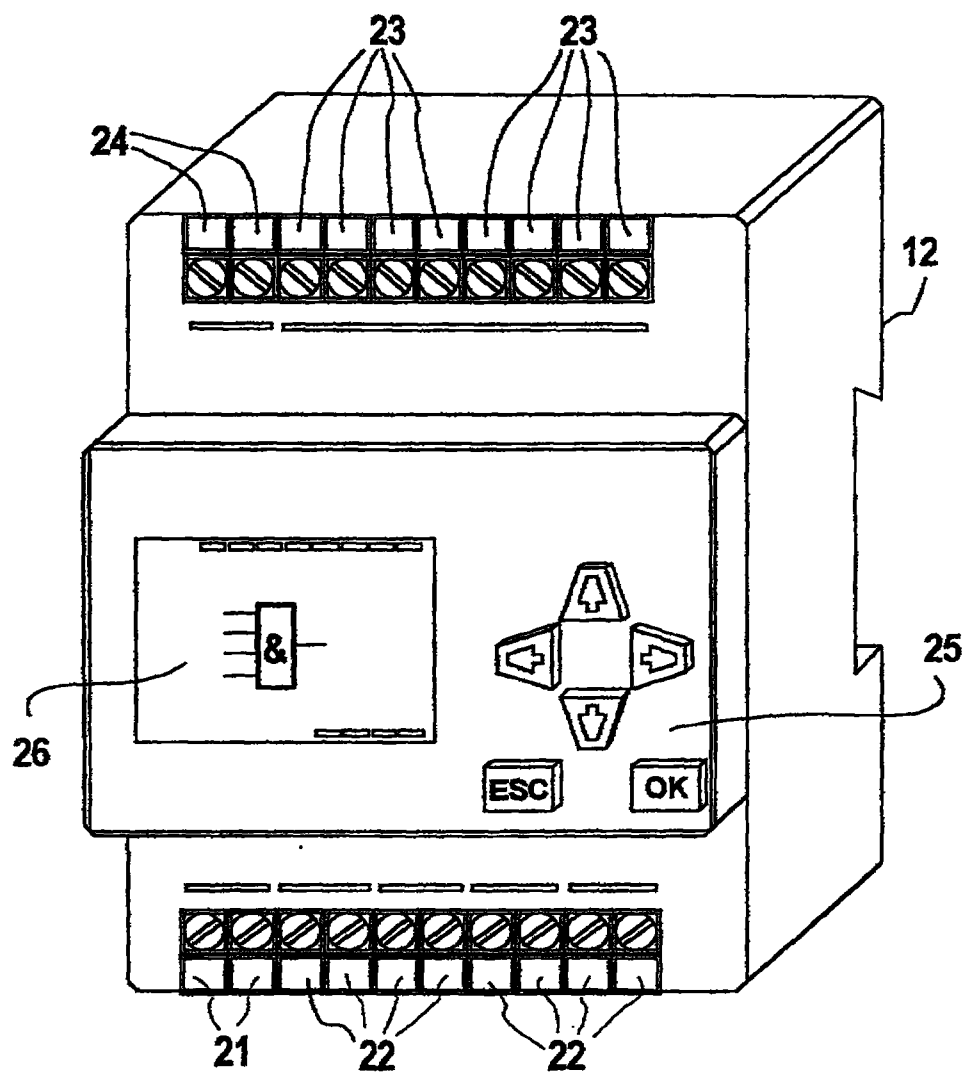


FIG 2

3/3

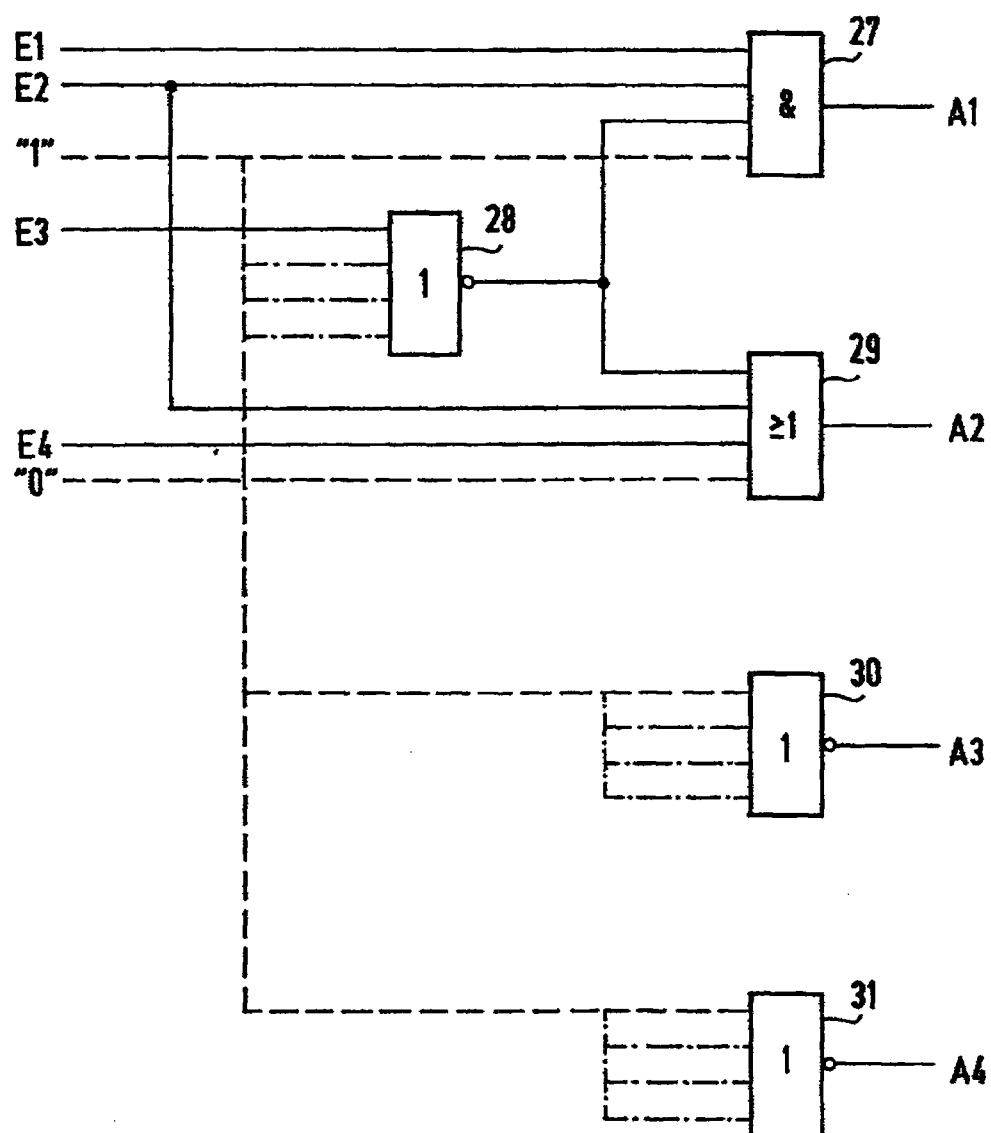


FIG 3

Description

Method for the generation and storage of a user program,
consisting of instructions, for a programmable con-
troller, and operating method for a programmable
5 controller

The present invention relates to a method for the
generation and storage of a user program, consisting of
instructions, for a programmable controller, the
programmable controller, when executing the user program,
10 determining output signals for an industrial process from
input signals from the industrial process, as well as to
an operating method, corresponding thereto, for a
programmable controller.

Methods of this type are generally known.

15 The object of the present invention is to make
new methods available.

The object is achieved for the generation method
and storage method in that the instructions of the user
program consist in each case of a function code and
20 function parameters assigned to the function code, in
that the function parameters refer either to an input
signal or to another instruction and in that the method
has the following steps:

- a) during the generation of the user program, for each
25 output signal, beginning with the instruction to be
assigned to the respective output signal, the
instructions to be executed are called up, the
following procedure being executed:
- b) the function code to be executed is called up and
30 stored temporarily;
- c) the number of function parameters is established;
- d) the function parameters are called up one after the
other;

- e) if the function parameter is an input signal, a reference to the input signal is stored temporarily;
 - f) if the function parameter is a new function code, a reference to a new instruction is stored temporarily, a branch is made to this new instruction and the other function code is stored temporarily and the steps c) to f) are repeated for the new instruction;
 - g) the temporarily stored function codes and the temporarily stored function parameters are stored in a nonvolatile memory and, respectively, identified as valid, only after all the instructions of at least the respective output signal have been called up completely.
- 15 In addition, according to the operating method, the instructions of the user program in each case consist of a function code and function parameters assigned to the function code, the function parameters referring either to an input signal or to another instruction. The
- 20 steps according to the invention are as follows:
- a) during the cyclic execution, for each instruction, beginning with the instructions assigned to the output signals, the following procedure is executed:
 - b) the function code assigned to the instruction and the function parameters assigned to this instruction are read;
 - c) for each function parameter a check is made as to whether this refers to another instruction;
 - d) if the function parameter refers to an input signal, the input signal is stored temporarily;
 - e) if the function parameter refers to another instruction, a branch is made to this other instruction and the steps b) to e) are repeated with the other instruction;
 - f) when all the function parameters have been exhausted, the output signal assigned to the instruction is determined.

Further advantages and details emerge from the following description of an exemplary embodiment. In this case:

FIG 1 shows the structure of a programming device and
5 a programmable controller,

FIG 2 shows the user interface of the programming device and

FIG 3 shows a programming example

According to FIG 1, the programming device 1 has
10 a processor 2 which is connected via the bus system 3 to the read-only memory 4 and the read-write memory 5 and to the interfaces 6, 7 and 8. The operating system of the processor 2 is stored in the read-only memory 4, the read-write memory 5 serving as working memory for the
15 programming device 1. Data can be entered into the programming device 1 via the interface 6 by means of the input unit 9, for example a keyboard. Data can likewise be displayed on the display unit 10 by means of the interface 7. By means of the communications interface 8,
20 the programming device is able to communicate with a programmable controller 12 via the communications connection 11 which is drawn with dashed lines.

The programmable controller 12 likewise has a processor 13, which executes a user program. The user
25 program consists of a multiplicity of instructions to be executed and is stored in the electrically eraseable and writeable read-only memory 14. In accordance with the user program to be executed, the processor 13 makes access via the controller 15 and the process interface 16
30 to an industrial process or to an industrial plant 17, which is not shown in more detail. From there, it reads in input signals, stores them in the read-write memory 18 and then determines output signals by reference to the user program. The output signals are likewise stored in
35 the read-write memory 18 and then output via the controller 15 and the process interface 16 to the

industrial process 17. In addition, the programmable controller 12 has an interface 19 to the programming device. The processor 13, the memories 14, 18, the controller 15 and the communications interface 19 are
5 likewise connected to one another via a bus 20.

In principle, programming device 1 and programmable controller 12 can be devices which are separate from each other. However, as is shown in FIG 1 by the dashed line and also shown in FIG 2, the programming device 1 is
10 preferably integrated into the programmable controller 12. However, it is possible in both cases for the programming device 1 to access the read-only memory 14 via the interfaces 8 and 19 and via the communications connection 11. In this arrangement it is possible for a
15 new user program to be written into the read-only memory 14 and for an existing user program to be read, erased or changed.

According to FIG 2, the programmable controller 12 has screw connections 21 to 24, a keyboard panel 25 and an LCD display 26. The programmable controller 12 can
20 be connected to a 24 volt power supply via the screw connections 21. Via two screw connections 22 in each case, it is possible to output a digital output signal to the industrial process 17. One input signal from the
25 industrial process 17 can be read in via each of the screw terminals 23. The actuators and sensors, which are connected to the screw terminals 22 and 23, can be supplied with power via the screw terminals 24. The screw terminals 22 to 24 are also an integral part of the
30 process interface 16.

The LCD display 26 corresponds to the display unit 10 of the programming device 1. In addition, messages from the programmable controller 12 can be displayed via the display 26 during operation. The display 26
35 therefore also serves as an observation unit. The keyboard panel 25 makes it possible to enter instructions

directly into the programming device 1 and into the programmable controller 12.

As is generally the case, the user program consists of individual instructions, each instruction
5 respectively consisting of a function code and function parameters assigned to the function code. In operation, the programmable controller executes the following sequence of instructions cyclically:

- It reads in input signals from the industrial process 17.
10
- It determines output signals for the industrial process by reference to the user program.
- It outputs the output signals to the industrial process 17.

15 This procedure is generally known.

During the generation of the user program, the instructions to be executed are called up one after another by the programming unit 1 from the input unit 9. For this purpose, beginning with the instruction to be
20 assigned to the first output signal, a query is raised as to which function is intended to be executed. A corresponding query is therefore output on the display unit 10 and the system waits until the user has input the function code via the input unit 9.

25 When the function code has been input, it is stored temporarily by the programming unit 1 in the read-write memory 5. The programming unit then establishes how many function parameters are assigned to this instruction. Here, a differentiation is to be made between the
30 internal assignment within the programming device 1 and the number of function parameters which are visible to the user. Internally, the number of function parameters is the same for all function codes, equal to four in the predetermined example. Externally, however, it is

always only the number of necessary function parameters which is called up. An example is intended to explain this in more detail:

5 If the code for the OR operation is entered as function code, in actual fact four parameters are called up. These four parameters are then also taken into account later during the execution of the program. However, if, for example, the logic negation is selected as function code, only one parameter, namely the input
10 signal to be negated, is called up from the user. The other three parameters are filled in automatically by the programming device 1 and not taken into account during the execution of the program.

Following the calling-up and temporary storage of
15 the function code in the read-write memory 5, the function parameters are also called up and temporarily stored one after another by means of the programming unit 1 working together with the input unit 9 and the display unit 10. Either an input signal or another function code
20 can be input as function parameter. In this case, when each function parameter is input, it is immediately checked as to whether this is a function code.

If the function parameter is not a function code but an input signal, this is stored temporarily and the
25 process is continued with the next function parameter. Input signals are in this case to be understood not only as input signals from the industrial process 17 but also signals whose value is known in advance. Examples of such signals are the constant one signal or the constant zero
30 signal, as well as a clock signal.

If, on the other hand, a new function code is input as function parameter, the function code is not stored temporarily as function parameter but the address of a new instruction. Immediately thereafter, a branch is
35 made to this new instruction and there the directly previously input function

code is stored temporarily. The number of function parameters is then reestablished and the function parameters are called up and stored temporarily.

5 In the case of these function parameters, too, a check is once more made immediately as to whether they are input signals or function codes. If they are new function codes, once more the address of this new instruction is stored immediately as function parameter and a branch is made to this new instruction. Here, too, the
10 function code which has just been input is stored once more, it is established how many function parameters are needed and the function parameters are called up and stored temporarily.

This nested calling up of function codes and
15 function parameters is continued until all the function parameters of the function code last called up and temporarily stored refer to input signals. When all the function parameters refer to input signals, the respective instruction has been completely input. A return is
20 then made to the preceding instruction and in the case of the latter, if appropriate, the next function parameter is called up and temporarily stored, a branch also being made here, if necessary, to further instructions.

The procedure mentioned above is continued until
25 all the functions necessary to determine an output signal have been nested inside one another, that is to say all the function parameters of each instruction have been input. Only when this has been carried out is the instruction assigned to the next output signal called up
30 and the procedure repeated, to the extent necessary. In this way, all the output signals are programmed one after another.

Of course, the input of the user program can be aborted at any time by the user. In this case, however,
35 no storage of the part-program which has been input and temporarily

stored is carried out. When the programming device 1 is switched off, the data which have been input are lost. Storage in the nonvolatile memory 14 of the user program which has been called up and temporarily stored in the read-write memory 5 is only made possible when the programming process has been fully completed. The programming process or the generation of the user program counts as fully completed

- when an instruction has been input for each output signal, and
- when each function parameter of each function code which has been programmed is occupied.

FIG 3 shows an example of a simple user program. In this case, the continuous lines signify the connections necessary on the basis of the desired linkages. The dashed lines show the additional connections required on the basis of the programming scheme, and the chain-dotted lines show the connections automatically added by the programming device 1. The input of the example program according to FIG 3 takes place as follows:

Firstly, the programming device asks, via display unit 10 and input unit 9, which function code is needed for the determination of the output signal of the first output A1. The user then inputs, via the input unit, the fact that an AND operation 27 is to be carried out. The corresponding function code is stored temporarily in the read-write memory 5 of the programming device 1. Then, one after another, the first, the second and the third function parameters, that is to say three of the four input signals of the AND operation 27, are called up and stored temporarily.

The first and the second input parameters are the input signals E1 and E2. Therefore, during the input of these two function parameters, the system continues immediately with the next function parameter. When the third function parameter is called up,

the user inputs the fact that he desires to undertake a NOT operation 28. As the third function parameter, therefore, the address of the instruction at which the NOT operation 28 will be handled, is temporarily stored.

5 A branch is then made by the programming device 1 to this instruction, and a query is raised as to which the (single) function parameter of the NOT operation 28 is intended to be. Here, the user inputs the input signal E3. The programming device 1 then stores this function
10 parameter temporarily and supplements the instruction with three further function parameters, which in each case refer to the logic one. These three function parameters are also stored temporarily in the memory 5. This then completes the setting of the parameters for the NOT
15 operation 28. A return is therefore made to the further setting of parameters for the AND operation 27, and the fourth function parameter is called up and temporarily stored. Since the AND operation 27 according to the
20 example of FIG 3 only has three true input parameters, the fourth input is connected to the logic one by means of a corresponding input by the user.

 This completes the programming of the output A1, and the process continues with the programming of the output A2.

25 Firstly, the query is raised and, following input by the user, the fact is temporarily stored that the function code for the generation of the output signal A2 is an OR operation 29. The four function parameters of the OR operation 29 are then called up one after another.

30 According to the programming example of FIG 3, the already programmed NOT operation of the input E3 is input as the first function parameter. Since this operation has already been

ready-programmed, no further input in relation to this function parameter is necessary and the process is continued with the second function parameter immediately after the temporary storage.

5 The second and the third function parameters are the (real) input signals E2 and E4, so that the process can continue immediately with the fourth function parameter.

10 According to the programming example of FIG 3, the OR operation 29 also has only three input signals. The fourth function parameter therefore refers to the constant zero.

 Since this also completes the programming of the output A2, the process continues with the output A3.

15 The outputs A3 and A4 are actually not needed in accordance with the given programming example. Therefore, the NOT operation 30 is selected for the output A3, which is linked to the logic one. As a result, the output A3 is connected to constant zero. The same takes place with the
20 output A4, for which the NOT operation 31 is input. As an alternative, it is also possible to connect the outputs A3 and A4 directly to logic zero.

 The program resulting therefrom is then stored in the memory 14. It is shown in the Table below.

25	B1:	FC	(AND)
		ADDR	(E1)
		ADDR	(E2)
		ADDR	(B5)
		ADDR	("1")
30	B2:	FC	(OR)
		ADDR	(B5)

```

        ADDR (E2)
        ADDR (E4)
        ADDR ("0")
    B3:  FC    (NOT)
5      ADDR ("1")
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")
    B4:  FC    (NOT)
10     ADDR ("1")
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")
    B5:  FC    (NOT)
15     ADDR (E3)
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")

```

20 On the left in Table 1, in each case the beginning of the individual instructions is represented by B1 to B5. This information, however, serves only for explanation and is not an integral part of the stored program. In the actual program, FC stands for function code, ADDR for address.

25 The program as such consists of blocks of five bytes each. The first byte in each case represents the function code of the instruction, the other four bytes represent the four function parameters. The first four instructions B1 to B4 always define the instructions for
30 the four output signals. Beginning at the fifth instruction B5, instructions are then stored to which access must be made indirectly during the calculation of the output signals.

35 The execution of the user program takes place as follows:

The instructions B1 to B4 are executed cyclically. During the execution of each of the instructions B1 to B4, the function code assigned to the instruction is read first. The function parameters assigned to this instruction are then read one after the other. During the reading of each function parameter, a check is immediately made as to whether this refers to another instruction. If the function parameter refers to an input signal, the latter is stored temporarily.

If, on the other hand, the function parameter refers to another instruction, a branch is made to this instruction immediately, that is to say before the reading of the other function parameters, and the function code assigned to this instruction and the function parameters assigned to this instruction are read. Here, too, if the function code points to a new instruction, this instruction is executed in turn, before the further function parameters are exhausted. When all the function parameters of an instruction have been exhausted, the output signal assigned to this instruction is determined, if appropriate a jump is made back to the calling instruction, and the determined output signal is used as a base for the further execution of the user program.

Here, too, it is intended to explain the principle in more detail with reference to the program according to the Table.

Firstly, therefore, the function code of the instruction B1 is read. The function parameters are then read, which refer to the inputs E1 and E2. Upon reading the third function parameter, a branch is made to the instruction B5. Its function code and all four function parameters (since all refer to input signals) are read and the output signal assigned to the instruction B5 is determined. A return is then made to the instruction B1 with this output signal and the fourth function parameter is read. The output signal A1 is then determined and stored in the read-write memory 18.

Then, the function code of the instruction B2 is read and the first function parameter of the instruction B2. A branch is then made once more to the instruction B5, whose function code and whose four function parameters are read and the output signal of the instruction B5 is determined. A return is then made to the instruction B2 and the other three function parameters are read. The output signal of the instruction B2 is then determined.

10 Finally, the function code and the function parameters of the instruction B3 are also read and executed and the output signal A3 thus determined. The same takes place for the instruction B4 and the output signal A4 resulting therefrom.

15 Following the determination of the last output signal A4, the output signals A1 to A4 thus determined are output to the industrial process 17 and the input signals E1 to E4 are read in again. A new cycle in the determination of the output signals A1 to A4 is then begun.

20 As an alternative to the creation method described, it is also possible to store the instructions which belong to an output signal immediately in the nonvolatile memory 14, if the respective output signal has been completely programmed. Therefore, for example, the instructions B1 and B5 could be stored immediately, as soon as the output signal A1 has been completely programmed.

25 Likewise, it is also possible to store each input function code and each input function parameter immediately in the nonvolatile memory 14. In this case, at the beginning of a program creation process, it would be necessary to store the fact that the programming of all the output signals A1 to A4 is invalid in an area of memory of the nonvolatile memory 14 specifically provided for this. For example, four 1-bit memory cells could be set to logic zero, in order to indicate that the four output signals A1 to A4 are not validly programmed.

The four 1-bit memory cells could then be set to logic one when the output signals A1 to A4 have been validly programmed. The as-valid identification can in this case be carried out optionally following the creation of the entire user program or separately for each output signal A1 to A4 following the calling up of all the instructions of the respective output signal A1 to A4.

The instructions also do not have to be stored in the order according to the Table. It is, for example, possible to store all the instructions which are needed for the determination of the output signal A1 here the instructions B1 and B5, one after the another in the memory 14. The instructions B2 to B4 for determining the output signals A2 to A4 would then be stored after these instructions (here, B1 and B5). During the execution of the user program it is necessary in this case to track which is the highest address which is called up for determining the output signal A1. The instruction for determining the output signal A2 then begins at the next highest address. In a similar way, namely temporary storage of the highest address needed, it is then possible to determine at which location the instructions for determining the output signals A3 and A4 are arranged. However, it is entirely up to the person skilled in the art which procedure is given preference in the implementation of the present invention.

Patent Claims

1. Method for the generation and storage of a user program, consisting of instructions (B1 to B5), for a programmable controller,
- 5 - the programmable controller, when executing the user program, determining output signals (A1 to A4) for an industrial process (17) from input signals (E1 to E4) from the industrial process (17),
- 10 - the instructions (B1 to B5) of the user program consisting in each case of a function code and function parameters assigned to the function code,
- the function parameters referring either to an input signal (E1 to E4) or to another instruction (B5),
- having the following steps:
- 15 a) during the generation of the user program, for each output signal, beginning with the instruction (B1 to B4) to be assigned to the respective output signal (A1 to A4), the instructions (B1 to B5) to be executed are called up, the following procedure
- 20 being executed:
- b) the function code to be executed is called up and stored temporarily;
- c) the number of function parameters is established;
- d) the function parameters are called up one after the
- 25 other;
- e) if the function parameter is an input signal (E1 to E4), a reference to the input signal (E1 to E4) is stored temporarily;
- f) if the function parameter is a new function code, a
- 30 reference to a new instruction (B5) is stored temporarily, a branch is made to this new instruction (B5) and the other function code is stored temporarily and the steps c) to f) are repeated for the new instruction (B5);
- 35 g) the temporarily stored function codes and the temporarily stored function parameters are stored in a



nonvolatile memory (14) and, respectively, identified as valid, only after all the instructions (B1 to B5) of at least the respective output signal (A1 to A4) have been called up completely.

- 5 2. Operating method for a programmable controller,
- the programmable controller cyclically executing a
user programme consisting of instructions (B1 to B5)
in order to determine output signals (A1 to A4) for
an industrial process (17) from input signals (E1 to
10 E4) from the industrial process (17),
- the instructions (B1 to B5) of the user program in
each case consisting of a function code and function
parameters assigned to the function code,
- the function parameters referring either to an input
15 signal (E1 to E4) or to another instruction (B5),
having the following steps:
a) during the cyclic execution, for each instruction
(B1 to B5), beginning with the instructions (B1 to
20 B4) assigned to the output signals (A1 to A4), the
following procedure is executed:
b) the function code assigned to the instruction (B1 to
B5) and the function parameters assigned to this
instruction (B1 to B5) are read;
c) for each function parameter a check is made as to
25 whether this refers to another instruction (B5);
d) if the function parameter refers to an input signal
(E1 to E4), the input signal (E1 to E4) is stored
temporarily;
e) if the function parameter refers to another instruc-
30 tion (B5), a branch is made to this other instruc-
tion (B5) and the steps b) to e) are repeated with
the other instruction (B5);



f) when all the function parameters have been exhausted, the output signal (A1 to A4) assigned to the instruction (B1 to B5) is determined.

3. Programming unit for implementing the method according to Claim 1, having

- means (2, 4, 5, 6, 7, 9, 10) for calling up and temporarily storing function codes of instructions to be executed;
- means (2, 4) for establishing the number of function parameters of the instructions to be executed;
- means (2, 4, 5, 6, 7, 9, 10) for calling up and temporarily storing the function parameters;
- means (2, 4) for checking the function parameters as to whether these are a new function code;
- means (2, 4) for branching to a new instruction if the function parameter is a new function code;
- means (2, 4) for preventing the storing of the temporarily stored function codes and the temporarily stored parameters in a nonvolatile memory (14) or, respectively, for preventing the as-valid identification if the function parameters of all instructions to be executed of an output signal (A1 to A4) have not yet all been stored temporarily.

4. Programmable controller for implementing the method according to Claim 2, having

- means (13) for reading function codes and function parameters assigned to instructions;
- means (13) for checking the function parameters as to whether these refer to an input signal or to another instruction;
- means (18) for storing the input signal temporarily if the function parameter refers to the input signal;
- means (13) for branching to the other instruction if the function parameter refers to the other instruction;



- means (13) for returning to a calling instruction;
- means (13) for determining the output signal assigned to the instruction when all the function parameters assigned to the latter have been exhausted.

5

5. Programmable controller according to Claim 4, which is combined with a programming unit according to Claim 3.



Abstract

Method for the generation and storage of a user program, consisting of instructions, for a programmable controller, and operating method for a programmable controller

During the program execution of a user program, the function code assigned to an output signal (A1 to A4) and the function parameters assigned to this function code are read first. For each function parameter, depending on whether this refers to an input signal (E1 to E4) or to another instruction (B5), either the input signal (E1 to E4) is stored temporarily or a branch is made to the other instruction (B1 to B5) and its function code and function parameters are executed. The generation of the user program is carried out in a similar way. In addition, however, in this case the storage or the as-valid identification of the user program is prevented if the function parameters of all the instructions (B1 to B5) of at least one output signal (A1 to A4) have not all been input.

FIG 3

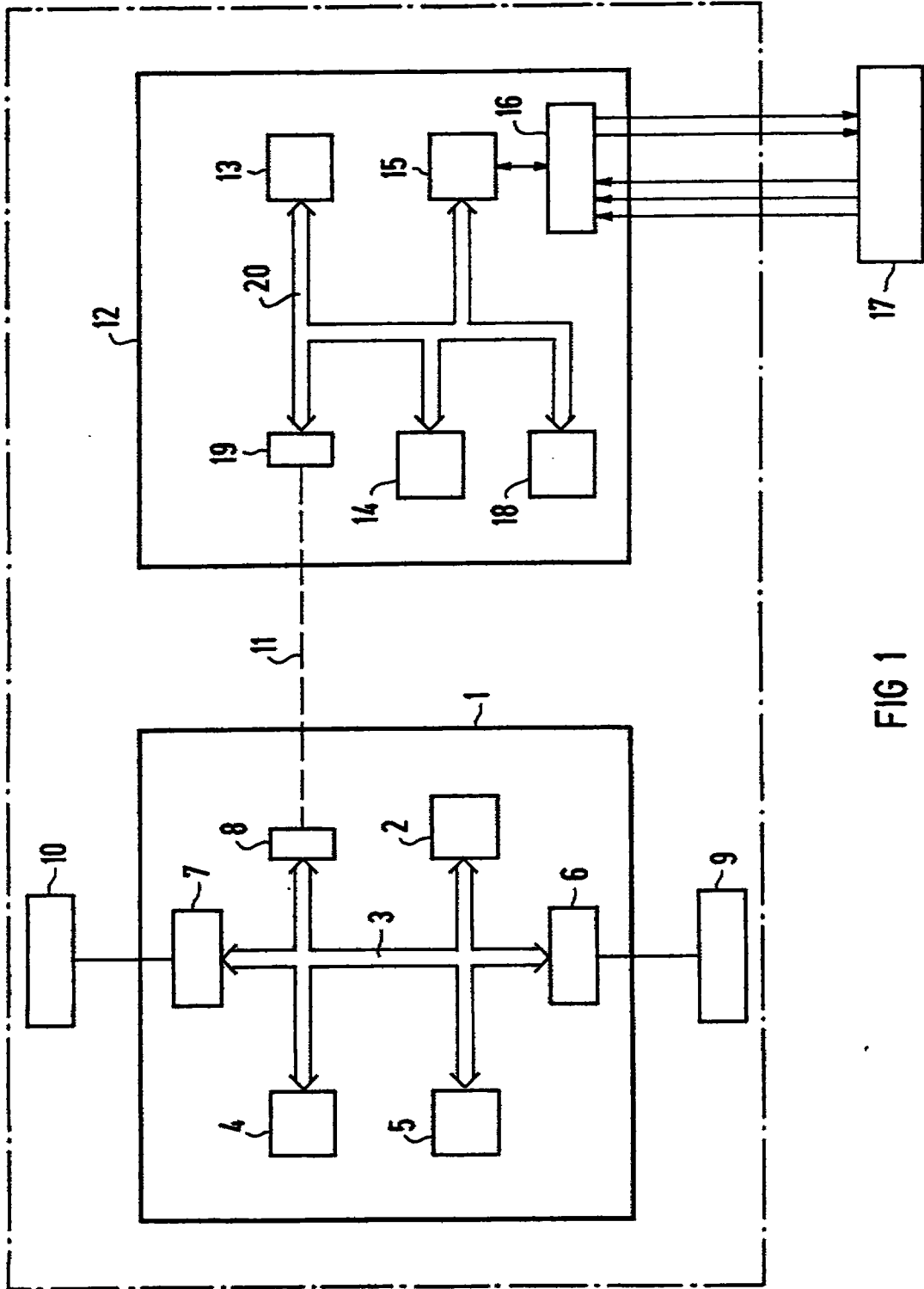


FIG 1

2/3

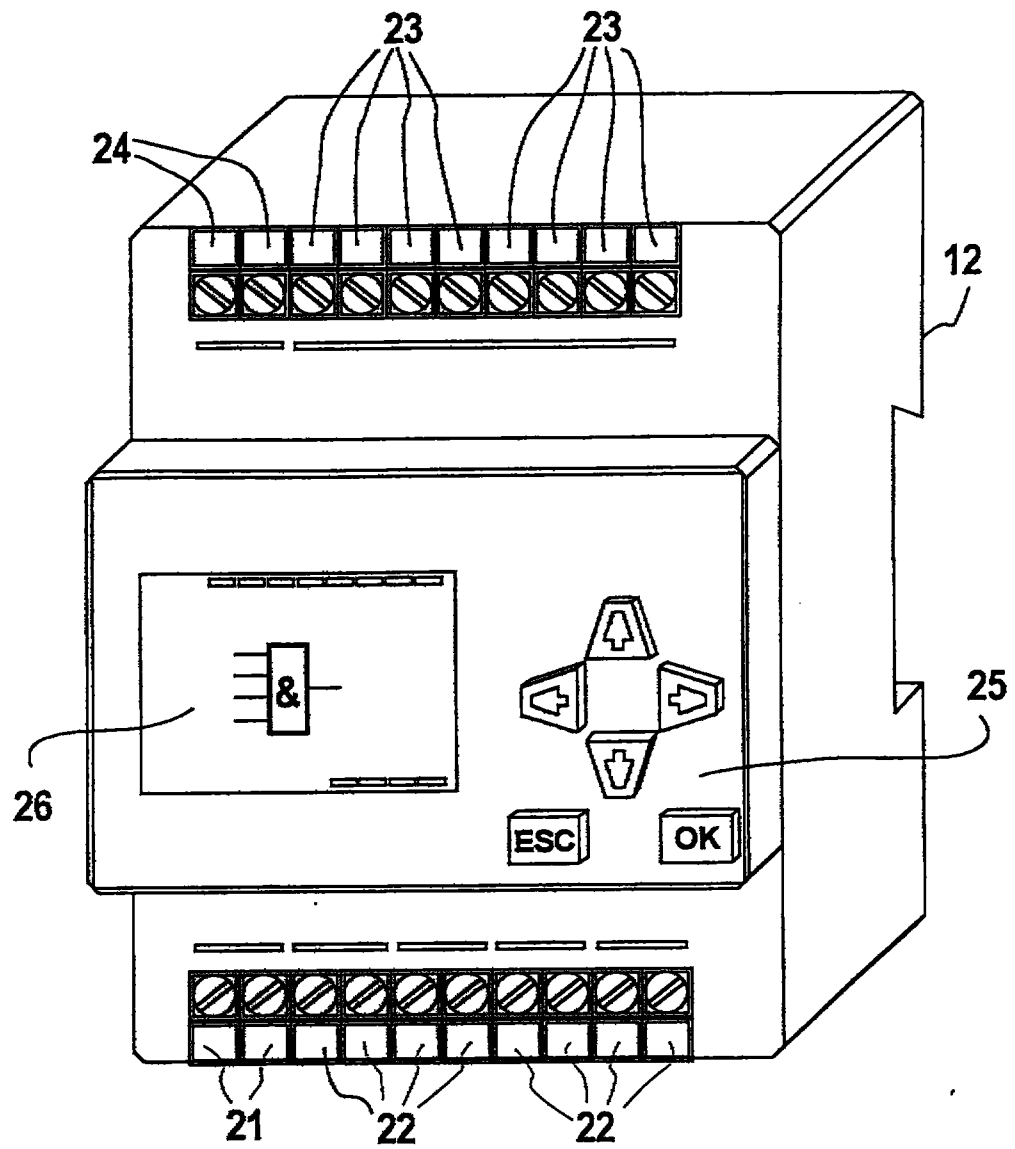


FIG 2

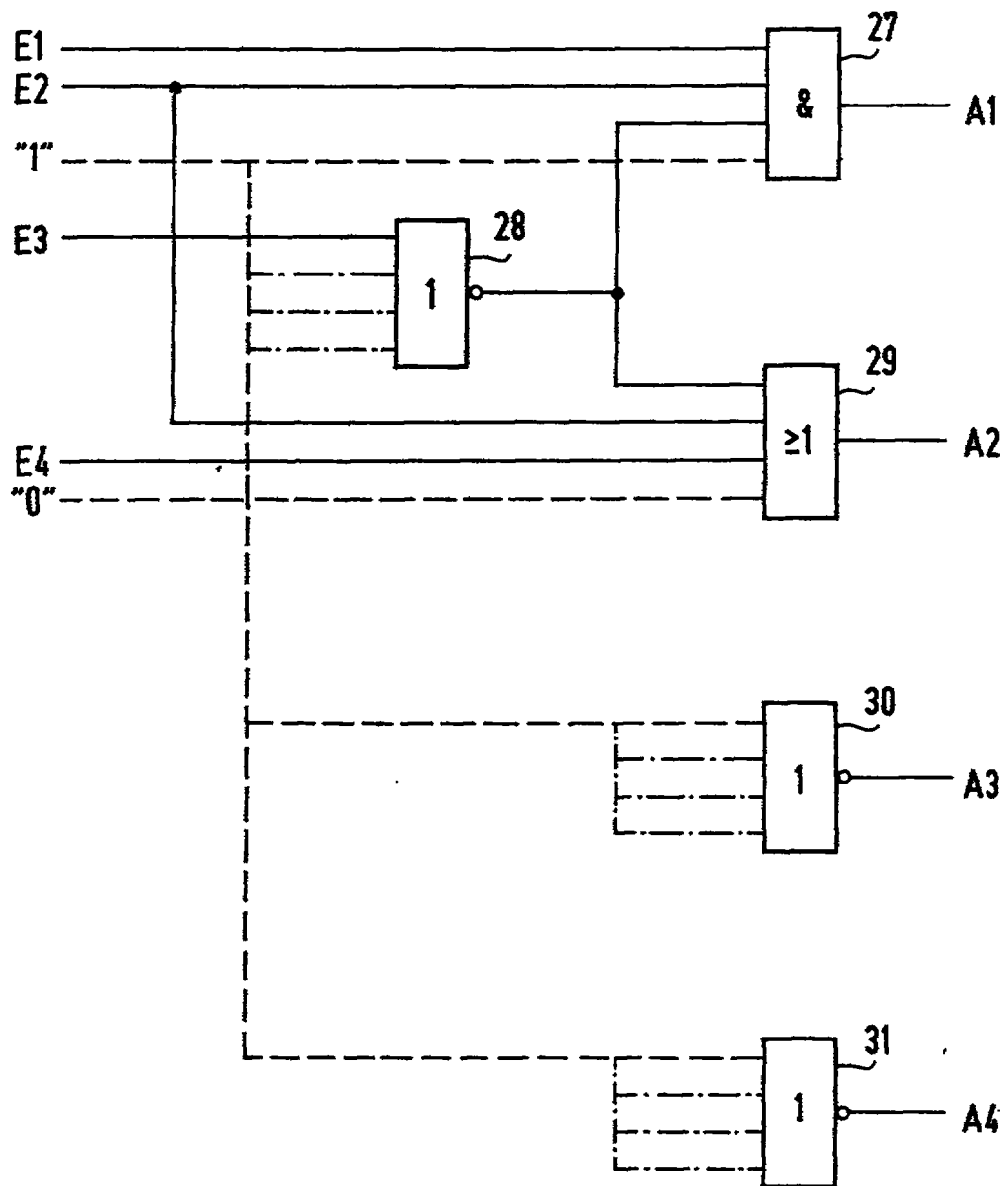


FIG 3



PCT
WELTORGANISATION FÜR GEISTIGES EIGENTUM
Internationales Büro
INTERNATIONALE ANMELDUNG VERÖFFENTLICHT NACH DEM VERTRAG ÜBER DIE
INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT)

<p>(51) Internationale Patentklassifikation ⁶ : G05B 19/05</p>	A1	<p>(11) Internationale Veröffentlichungsnummer: WO 97/02661</p> <p>(43) Internationales Veröffentlichungsdatum: 13. März 1997 (13.03.97)</p>
<p>(21) Internationales Aktenzeichen: PCT/EP96/03744</p> <p>(22) Internationales Anmeldedatum: 23. August 1996 (23.08.96)</p> <p>(30) Prioritätsdaten: 95113939.3 / 5. September 1995 (05.09.95) EP</p> <p>(34) Länder für die die regionale oder internationale Anmeldung eingereicht worden ist: DE usw.</p> <p>(71) Anmelder (für alle Bestimmungsstaaten ausser US): SIEMENS AKTIENGESELLSCHAFT (DE/DE); Wittelsbacherplatz 2, D-80333 München (DE).</p> <p>(72) Erfinder; und</p> <p>(75) Erfinder/Anmelder (nur für US): KAMMLER, Heiko (DE/DE); Goethestrasse 17, D-92284 Popenricht (DE). SCHIESSL, Michael (DE/DE); Crayerstrasse 9, D-92224 Amberg (DE).</p>	<p>(81) Bestimmungsstaaten: CN, CZ, HU, JP, KR, PL, <u>SG</u>, US, europäisches Patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).</p> <p>Veröffentlicht <i>Mit internationalem Recherchenbericht.</i></p>	

(54) Title: METHOD OF GENERATING AND STORING A USER PROGRAMME CONSISTING OF INSTRUCTIONS FOR A STORED-PROGRAMME CONTROL UNIT, AND METHOD OF OPERATING THE STORED-PROGRAMME CONTROL UNIT

(54) Bezeichnung: VERFAHREN ZUM ERZEUGEN UND ABSPEICHERN EINES AUS BEFEHLEN BESTEHENDEN ANWENDER-PROGRAMMS FÜR EINE SPEICHERPROGRAMMIERBARE STEUERUNG UND BETRIEBSVERFAHREN FÜR EINE SPEICHERPROGRAMMIERBARE STEUERUNG

(57) Abstract

When a user programme is executed, the invention calls for the function code associated with an output signal (A1 to A4), and the function parameters associated with this function code, to be read. Per function parameter, depending whether the parameter refers to an input signal (E1 to E4) or to another instruction (B5), either the input signal (E1 to E4) is temporarily stored or branched off to the other instruction (B1 to B5), and its function code and function parameter executed. Generation of the user programme is carried out in analogous fashions. In addition, however, storage or identification as valid of the user programme is prevented if not all of the function parameters for all the instructions (B1 to B5) for at least one output signal (A1 to A4) have been input.

(57) Zusammenfassung

Bei der Programmabarbeitung eines Anwenderprogramms wird zunächst der einem Ausgangssignal (A1 bis A4) zugeordnete Funktionscode und die diesem Funktionscode zugeordneten Funktionsparameter gelesen. Pro Funktionsparameter wird, je nachdem, ob dieser auf ein Eingangssignal (E1 bis E4) oder auf einen anderen Befehl (B5) verweist, entweder das Eingangssignal (E1 bis E4) zwischengespeichert oder zu dem anderen Befehl (B1 bis B5) verzweigt und dessen Funktionscode und Funktionsparameter abgearbeitet. Die Erzeugung des Anwenderprogramms erfolgt analog. Zusätzlich wird dabei aber das Abspeichern bzw. das Als-gültig-Kennzeichnen des Anwenderprogramms verhindert, wenn nicht alle Funktionsparameter aller Befehle (B1 bis B5) zumindest eines Ausgangssignals (A1 bis A4) eingegeben sind.

