Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2835535 A1 2012/11/22

(21) 2 835 535

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2011/05/13
(87) Date publication PCT/PCT Publication Date: 2012/11/22
(85) Entrée phase nationale/National Entry: 2013/11/08
(86) N° demande PCT/PCT Application No.: EP 2011/057769
(87) N° publication PCT/PCT Publication No.: 2012/155949

(51) Cl.Int./Int.Cl. *G05B 19/418* (2006.01), *G05B 19/042* (2006.01), *H04L 12/28* (2006.01), *H04L 29/08* (2006.01)

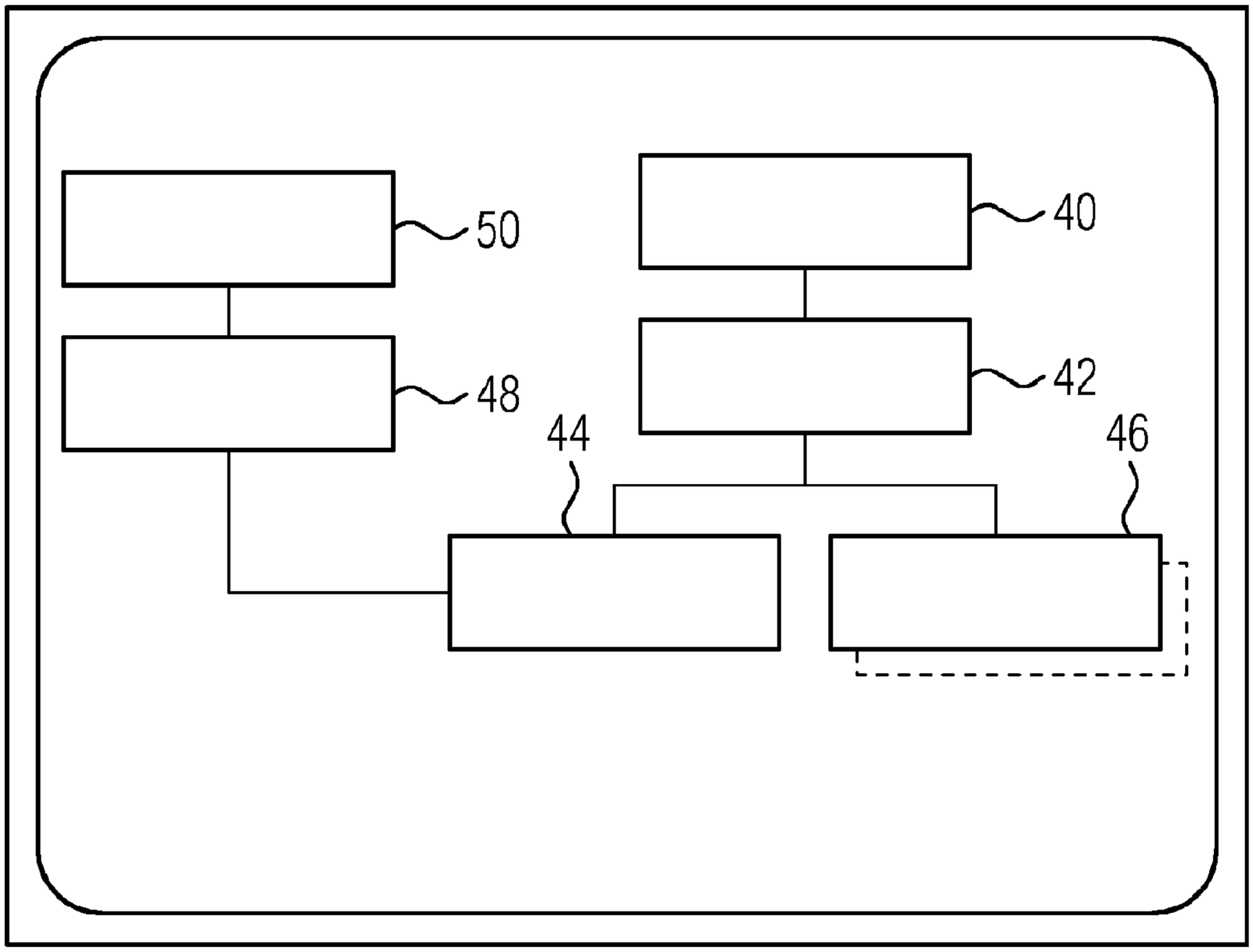
(71) Demandeur/Applicant:
SIEMENS AKTIENGESELLSCHAFT, DE

(72) Inventeur/Inventor:
WEISS, HERBERT, DE

(74) Agent: SMART & BIGGAR

(54) Titre : PROCEDE PERMETTANT DE FAIRE FONCTIONNER UN SYSTEME D'AUTOMATISATION
(54) Title: METHOD FOR OPERATING AN AUTOMATION SYSTEM

FIG 3



(57) Abrégé/Abstract:
The invention relates particularly to a method for operating an automation system (10), wherein the automation system (10) comprises, in communicatively connected form, a superordinate IO link unit (14) and at least one modular IO link appliance (20)

(21) 2 835 535

(13) A1

(57) Abrégé(suite)/Abstract(continued):
having an appliance-internal bus (32) and subunits (24, 26, 28) which can be addressed by means of the latter and which the modular IO link appliance (20) comprises, wherein the method is distinguished in that communication with the modular IO link appliance (20) involves the selection of one of the subunits (24, 26, 28) thereof, and the communication takes place only with this subunit directly and, via the latter with the other subunits (24, 26, 28) of the modular IO link appliance (20) indirectly.

Abstract

The invention relates particularly to a method for operating an automation system (10), wherein the automation system (10) comprises, in communicatively connected form, a superordinate IO link unit (14) and at least one modular IO link appliance (20) having an appliance-internal bus (32) and subunits (24, 26, 28) which can be addressed by means of the latter and which the modular IO link appliance (20) comprises, wherein the method is distinguished in that communication with the modular IO link appliance (20) involves the selection of one of the subunits (24, 26, 28) thereof, and the communication takes place only with this subunit directly and, via the latter with the other subunits (24, 26, 28) of the modular IO link appliance (20) indirectly.

Description

Method for operating an automation system

The invention relates to a method for operating an automation system, particularly an automation system comprising IO-Link devices, and to a method for managing such IO-Link devices, in particular with regard to configuration and parameterization.

A concept for the standardized linking of sensors and actuators (e.g. switchgear) to a control level by means of a low-cost point-to-point connection is known under the IO-Link trade mark registered to the PROFIBUS User Organization (PROFIBUS Nutzerorganisation e.V.). This communication standard below the fieldbus level enables central error diagnosis and location down to the sensor/actuator level. As an open interface, IO-Link can be integrated into all common fieldbus and automation systems. The above-mentioned communication system will be referred to below in abbreviated form as IO-Link.

The IO-Link specification (current version: V1.0, 2008/2009) describes how IO-Link devices from different equipment manufacturers can be linked to a point-to-point connection. In accordance with the specification, parameters, diagnoses, etc. for these devices can be transferred from and to a so-called IO-Link master as a superordinate IO-Link unit. The term 'diagnosis' is understood here and hereinbelow to refer on the one hand to diagnostic information resulting from a check or a status query of the particular device and on the other to a description of a type and/or scope of such a check or status query, as well as to measured values (currents, voltages, temperatures, etc.), statistical data (hours of operation,

etc.), logbooks, etc. The IO-Link devices, in particular the respective parameters, diagnoses, etc., are described in a dedicated device description file (IODD).

However, the device description does not permit modeling of modular IO-Link devices such as e.g. the so-called compact starters offered by the applicant under the name "SIRIUS 3RA6". Only compact IO-Link devices can be described. Information on modularity is, as it were, hidden device-specifically in parameters or diagnostic information (e.g. error messages, service life, end position, etc.) that are not needed or are less relevant.

Due to this restriction, modular IO-Link devices can be represented in the configuration and diagnosis of IO-Link engineering software (e.g. SIMATIC Step7) only as a compact device with universal configuration, diagnosis, etc. Such representations are thus often misleading or even incorrect. For example, a central group-error LED provides no information as to which of a plurality of subunits of a modular IO-Link device is faulty.

It is also not possible, in particular,

- to select the individual subunits, i.e. modules or devices, of a modular IO-Link device from a hardware selection catalog,
- to configure the modular IO-Link device by means of standard user actions such as drag & drop, either in graphic or in tabular form,
- to represent the actual device configuration offline and online,
- to carry out a target/actual comparison of the device configuration,

- to represent the diagnostic displays (e.g. device LEDs) of the individual modules/devices in the online configuration,
- to keep the sizes of the process images, i.e. the process image of the inputs (PAE) and process image of the outputs (PAA), consistent,
- to calculate the address length and assignment automatically depending on the expansion or
- to display a product image that corresponds to the actual expansion.

A further disadvantage is that two different development tools are always required for IO-Link engineering, namely a first development tool, e.g. the applicant's engineering software known under the name STEP7, for configuring the IO-Link master in the automation system and a second development tool for configuring the IO-Link master itself and the IO-Link devices communicatively connected thereto.

The configuring of an IO-Link system as an integral part of an automation system using the applicant's development tools, i.e. on the one hand the development tool known under the name SIMATIC Step7 (first development tool) and on the other the development tool known under the name Port Configuration Tool (S7-PCT) (second development tool), will be described below for illustrative purposes.

The following steps should be executed when configuring an IO-Link system in the engineering software:

1) Configuration of the IO-Link master in the first development tool, e.g. by dragging & dropping an IO-Link master into the hardware configuration of the first development tool, e.g.

into an automation device created there in the form of a programmable logic controller or similar.

2) Parameterization of the IO-Link master in the first development tool. Here, the user enters the I/O addresses (start and length) and the diagnostic parameters of the IO-Link master. To determine the length of the I/O addresses, the user must know the exact number and type of the ports and/or IO-Link devices used. In modular IO-Link devices, the problem is exacerbated by the fact that the size of the process images of the inputs and outputs depends in turn on the modules/devices used. There is no guarantee that the first development tool will support this, as in modular IO-Link devices the subunits which the devices comprise are not known. As a result, address assignment is error-prone.

3) Configuration of the IO-Link devices in the second development tool by calling the second development tool, e.g. directly from the first development tool, and by dragging & dropping the IO-Link devices into a port configuration which the second development tool comprises. For this purpose, IO-Link devices, where available, can be integrated using a device description file.

4) Parameterization of the IO-Link devices in the second development tool by selecting the relevant Port/IO-Link device and inputting the device parameters. In modular IO-Link devices, the configuration of subunits which they comprise is input "hidden" in the device parameters. The individual modules/devices of the modular IO-Link device - referred to hereinbelow as a subunit or IO-Link subdevice - cannot be selected in the hardware selection catalog. Nor is

it possible to configure them graphically by dragging & dropping the individual subunits.

5) Downloading of the parameters into the IO-Link master and the individual IO-Link devices by means of the first development tool, based on prior transfer of the parameters of the IO-Link devices in conjunction with termination of the second development tool. The parameters are stored in the data store of the first development tool. During the download, the device parameters are loaded onto a central unit of an automation device which then transfers these on startup to the IO-Link master and to the IO-Link devices.

6) Diagnosis of the IO-Link system in the first development tool by reading out and displaying the system diagnostic information of all accessible modules. In the case of the IO-Link master, this is the group diagnostic information for the master (matches the status of a dedicated IO-Link master LED) and the diagnostic information for the ports / IO-Link devices. In modular IO-Link devices, this individual diagnostic information cannot represent the possibly different states of the LEDs of the individual IO-Link subdevices (subunits; feeders).

7) In order to obtain exact diagnostic information in modular IO-Link devices, the user must switch to the second development tool and view the device diagnosis there. Here, in the case of modular IO-Link devices, the status of the LEDs is represented by the obtainable diagnostic information (group error, group warning, etc.). Besides diagnostic information, the inputs/outputs of the individual subunits can be viewed.

A first object of the approach proposed here is to simplify the management of modular IO-Link devices and their use in an automation system.

This object is achieved in a method having the features summarized in claim 1. To this end, a method for operating an automation system, the automation system comprising, in communicatively connected form, a superordinate IO-Link unit, e.g. an IO-Link master, and at least one modular IO-Link device having a device-internal bus and subunits which can be addressed by means of the latter and which the modular IO-Link device comprises, provides that communication with the modular IO-Link device involves the selection of one of the subunits thereof, and communication takes place directly only with this subunit and indirectly via the latter with the other subunits of the modular IO-Link device.

A further object of the approach proposed here is, given the scope and the complexity of the method steps hitherto required for configuring and parameterizing modular IO-Link devices, to simplify the management of said method steps for configuration, parameterization and/or diagnosis.

A modular IO-Link device is a "pseudomodular compact device", i.e. it comprises a plurality of modules referred to here as subunits or feeders (IO-Link subdevices), which are connected via a device-internal bus. As an example, reference can be made to the device offered by the applicant under the name “SIRIUS 3RA6 compact starter”. Until now, however, the individual subunits have not been visible externally and have not been directly addressable, i.e. because they have no address, are not assigned to a port in the IO-Link model, provide no diagnostic information, etc.

The problems outlined above can be eliminated by means of the following technical features:

1) Extension of an IO-Link object model to include subunits (IO-Link subdevices)
2) Modeling of such subunits in the device description (e.g. IODD)
3) Graphical or tabular representation of the modularity of modular IO-Link devices in a single development tool.

The further object is achieved correspondingly in a method having the features summarized in claim 2. To this end, an automation system or a method for configuring or parameterizing a modular IO-Link device in such an automation system provides that, when a modular IO-Link device is created in a development tool, a first object is created to represent the modular IO-Link device, a second object is created to represent an IO-Link rack in the modular IO-Link device, said rack comprising or accommodating the subunits, a third object is created to represent the subunit selected and functioning as an IO-Link header module and at least one fourth object is created for each further subunit of the modular IO-Link device. The first, second, third and fourth object or the object types on which each is based represent the extension of an IO-Link object model. The creation of an object to represent the modular IO-Link device and of any further objects is carried out using the development tool for an automation solution for controlling and/or monitoring a technical process. The automation solution comprises the automation system and thus also the IO-Link system as an integral part of the automation system having at least one IO-Link master and one modular IO-Link device but also software for defining the functionality of the individual units of the

automation system and their configuration, parameterization, etc.

The solution proposed here is based on one of the subunits taking over outward communication via IO-Link and thereby functioning, as it were, as a proxy for the entire modular IO-Link device. This subunit is also referred to below as a header module, in order to differentiate it. The header module may be a specified subunit, e.g. a communication module, or any subunit of the modular IO-Link device that manages the communicative connection of the modular IO-Link device and thereby functions as a header module. Only this header module needs information regarding the internal structure of the IO-Link device. On this basis, the subunits which the IO-Link device comprises are addressed by the header module via an entirely internal mechanism. The presence of the subunits impacts only on the technological functions of the modular IO-Link device, by enabling or activating in another suitable manner the functions of these subunits (by parameters, diagnosis, process images, etc.).

The extension of the IO-Link object model with such subunits is necessary for this reason. The extended object model allows a modular IO-Link device to be handled on the user interface of a single development tool as a modular device with a plurality of subunits. All the subunits which the modular IO-Link device comprises can thus be selected in the development tool and added in the device view to the modular IO-Link device, e.g. by dragging & dropping them.

The key special characteristics of the extended object model and the objects provided for it are described below. The basic assumption of the extended object model is that a modular IO-

Link device is always modeled by a combination of the following objects: a (first) object for an IO-Link device, a (second) object for an IO-Link rack, a (third) object for an IO-Link header module and at least one (fourth) object for a subunit of the modular IO-Link device. Reference will not always be made below to an object as the representation of a physical unit, so instead of expressions that are complete in themselves, such as "object for modeling the IO-Link device", for example, short forms such as "IO-Link device" may be written. It will be evident from the context in each case whether an object is meant as the representation of a physical unit or as the physical unit itself.

Advantageous embodiments of the approach are the subject of the dependent claims. References used here refer to further development of the subject matter of the main claim by the features of the respective dependent claim; they are not to be understood as abandoning the aim of achieving independent, objective protection for the combinations of features of the dependent claims referred to. Furthermore, with regard to interpretation of the claims, where there is a more detailed definition of a feature in a subordinate claim, it must be assumed that no restriction of this kind is present in any of the preceding claims.

The object for modeling the IO-Link device is that object which in a master view (IO-Link master view) is displayed to represent the IO-Link device. The object for modeling the IO-Link rack is a container for the subunits which the modular IO-Link device comprises and which can thus be configured in the device view of the development tool. First slot rules concerning the relations of the object for modeling the IO-Link rack to the subunit or to each subunit are mapped, i.e.

such slot rules as can be checked when an object for modeling a subunit is combined with the IO-Link rack.

The object for modeling the header module functions as a proxy for a modular IO-Link device. It represents the actual technological functionality of the modular IO-Link device and carries most of the device's characteristics (device parameters, length of the input and output address, device diagnoses, etc.). Modular IO-Link devices are pseudo-modular systems, i.e. they can be configured with subunits. Unlike true modular systems, however, they do not have their own parameters and accordingly have no contact data of their own. Instead, a subunit changes the characteristics of the modular IO-Link device and the characteristics of the object modeling the modular IO-Link device, e.g. the visibility of parameters. For this, the header module must have information about which subunits are currently configured. This information can be determined via object references from the device description.

The header module is the only object for which a physical counterpart exists in each case. The header module represents the modular IO-Link device on the user interface of the development tool. It is therefore also the object which is designated by e.g. an order number (MLFB) of an IO-Link device and can thus be generated by the user via the hardware catalog. All other objects (IO-Link rack, IO-Link device, IO-Link subunit) are then implicitly additionally generated.

The actions and dialogs relevant for parameterization, diagnosis, address allocation, etc. can be accessed via the header module, i.e. the header module object. The header module object additionally comprises the functionality for integrating individual or multiple subunits and the

functionality for connecting to the IO-Link master, i.e. a dedicated IO-Link port (node). Objects for managing address information (address objects) and the IO-Link port (node) are standard objects and are generated automatically where the device description contains appropriate attributes, as is the link to the IO-Link master system. Apart from an icon in the device view and the standard dialogs for project information, a subunit of a modular IO-Link device does not have its own user interface, its own device parameters or any links to the IO-Link system.

One embodiment of the method provides accordingly that, when an object is created for a modular IO-Link device, an object is automatically created for the selected subunit of the modular IO-Link device functioning as an IO-Link header module and/or an object is created for the IO-Link rack of the modular IO-Link device, and in particular, and likewise automatically, an interconnection between these objects is created. This, on the one hand, reduces method steps that would otherwise be required manually and, on the other hand, ensures the consistency of representation of the automation system in the development tool by ensuring that in a modular IO-Link device the connection to IO-Link is effected exclusively via the subunit selected as the header module and by further ensuring that proxies are available for the IO-Link device itself, which comprises the header module, and for the IO-Link rack.

The approach described here also comprises modeling of IO-Link subunits in the device description. Currently - i.e. in accordance with the prior art - only the IO-Link device and its characteristics (parameters, diagnoses, etc.) are

described in the device description. Any IO-Link subunits are, as far as possible, covered by device parameters.

Inserted below is an example of a prior-art device description for the above-mentioned device of the applicant, namely the SIRIUS 3RA6 compact starter:

```
<Text id="TI_DS130_Byte12" value="Starter type" />
<Text id="TI_DS130_BGID_DS_010-040"
   value="Direct starter DS 0,1 ... 0,4 A" />
<Text id="TI_DS130_BGID_DS_032-125"
   value="Direct starter DS 0,32 ... 1,25 A" />
<Text id="TI_DS130_BGID_DS_100-400"
   value="Direct starter DS 1 ... 4 A" />
<Text id="TI_DS130_BGID_DS_300-1200"
   value="Direct starter DS 3 ... 12 A" />
<Text id="TI_DS130_BGID_DS_800-3200"
   value="Direct starter DS 8 ... 32 A" />
<Text id="TI_DS130_BGID_RS_010-040"
   value="Reversing starter RS 0,1 ... 0,4 A" />
<Text id="TI_DS130_BGID_RS_032-125"
   value="Reversing starter RS 0,32 ... 1,25 A" />
<Text id="TI_DS130_BGID_RS_100-400"
   value="Reversing starter RS 1 ... 4 A" />
<Text id="TI_DS130_BGID_RS_300-1200"
   value="Reversing starter RS 3 ... 12 A" />
<Text id="TI_DS130_BGID_RS_800-3200"
   value="Reversing starter RS 8 ... 32 A" />
```

For modeling a modular IO-Link device with IO-Link in accordance with the approach proposed here, both the object model and the device description are extended to include at least

- a description of all the objects of the object model, namely IO-Link device, IO-Link rack, IO-Link header module and IO-Link subunit,
- description of the device configuration: number of slots, pluggable device types, etc.
- usable types of IO-Link subunits,
- description of the IO-Link subunit for its representation in the hardware catalog and
- slot rules.

With regard to the slot rules, it is possible to cover only "hard" slot rules, hard in this context meaning that selection of an IO-Link subunit and its combination with the modular IO-Link device from the hardware catalog is prevented.

Only the most important attributes that are required for mapping the extended IO-Link object model in the device description are described below. Where specific values are listed by way of example, these are often values for the previously mentioned device, namely the SIRIUS 3RA6 compact starter. The device parameters and diagnoses can be described here in the same way as is already currently provided for in the device description (byte offset, bit offset, data type, length, default value, ID, etc.).

General attributes of all objects:

```
VARIABLE Comment;
   // Device-specific comment,
   // e.g. "Conveyor belt Hall 3, Motor 12"
VARIABLE Name;
   // Device-specific name,
   // e.g. "Conveyer23.Starter07"
```

```
VARIABLE MLFB;
   // Order number, e.g. "3RA6234-0AA1-0BBX"
VARIABLE ObjectType;
   // Type of object
VARIABLE ObjectId;
   // Unique identification number of the object
```

Attributes for objects for modeling an IO-Link device:

```
VARIABLE ObjectType;
   // Type of object = IO_LINK_DEVICE
   // (unique object type number, constant)
VARIABLE ObjectId;
   // Identification number of the object,
   // e.g. SIRIUS_3RA6 (unique object number, constant)
VARIABLE ContainerSize: 1;
   // an IO-Link device always includes a rack
VARIABLE TypeName;
   // for display in the development tool,
   // e.g. "SIRIUS 3RA6 compact starter"
```

Attributes for objects for modeling an IO-Link rack:

```
VARIABLE ObjectType;
   // Type of object = IO_LINK_DEVICE_RACK
   // (unique object type number, constant)
VARIABLE ObjectId;
   // Identification number of the object, e.g.
   // SIRIUS_3RA6_3RA6 (unique object number, constant)
VARIABLE ContainerSize: 4;
   // Number of available slots in the rack;
   // in the SIRIUS 3RA6 compact starter e.g. 4
```

Attributes for objects for modeling an IO-Link header module:

```
VARIABLE ObjectType;
   // Type of object = IO_LINK_HEAD_DEVICE
   // (unique object type number, constant)
VARIABLE ObjectId;
   // Identification number of the object, e.g.
   // SIRIUS_3RA6_HEAD (unique object number, constant)
VARIABLE PositionNumber;
   // The current slot number of the header module, in the
   // SIRIUS 3RA6 compact starter, from 0 to 3
VARIABLE UICapabilities: 0;
   // 0 = the module is virtual and is not
   // displayed in the development tool
VARIABLE InAddressRange;
VARIABLE OutAddressRange;
   // Size of the input and output addresses of the slave in
   // the address space of the SPS / IO-Link master system
   // e.g. in the compact starter: 8 = 8 bytes, 16 = 16 bytes
VARIABLE PARAMETER_17;
   // A device parameter of the IO-Link device, analogous to
   // the IODD specification
VARIABLE DIAGNOSIS_27;
   // A device diagnosis of the IO-Link device,
   // analogous to the IODD specification
```

Attributes for objects for modeling an IO-Link subunit:

```
VARIABLE ObjectType;
   // Type of object = IO_LINK_SUB_DEVICE
   // (unique object type number, constant)
VARIABLE ObjectId;
   // Identification number of the object, e.g.
```

```
   // SIRIUS_DIRECTSTARTER_3_12A
   // (unique object number, constant)
VARIABLE PositionNumber:
   // the current slot number of the module, in the
   // SIRIUS 3RA6 compact starter, from 0 to 3
VARIABLE TypeCode;
   // including for parameterization. e.g.
   // Bit 15 == 1
   //      (Module is not parameterizable)
   // Bit 15 == 0
   //      (Module is parameterizable)
VARIABLE FWMainVersion: 1;
VARIABLE FWVersion: "V1.0";
   // Firmware version
VARIABLE UICapabilities: 1;
   // 1 = The module is displayed in the development tool
```

Based on the extended IO-Link object model described here and the device description, also extended, that maps the extended IO-Link object model, a modular IO-Link device can be represented in the development tool with all the subunits which it comprises. The representation comprises at least one option for diagnosing individual IO-Link subunits with a display of the results of the diagnosis. The representation then comprises a display of an expansion and/or a configuration of the modular IO-Link device with its subunits, and does so in tabular and/or graphic form. Finally the representation also comprises support of the hardware catalog for device selection.

Due to the extension of the object model and device description for IO-Link and of the graphic representation in the development tool, it is now possible at least

- to select the individual subunits of a modular IO-Link device from a hardware selection catalog,
- to configure the modular IO-Link device by dragging & dropping (in graphic or tabular form),
- to represent an actual device configuration offline and online respectively,
- to perform a target/actual comparison of the configuration,
- to represent diagnostic information obtained or obtainable from the individual subunits on the respective subunit in the hardware configuration,
- to keep the sizes of the process images of the inputs and outputs (PAE and PAA, respectively) consistent,
- to calculate the address length and assignment automatically depending on the expansion,
- to display a product image that corresponds to the actual expansion and
- to display full and correct customer documentation.

This makes the overall configuration of an IO-Link system more efficient for the customer and less prone to errors. Fewer steps for configuration are needed overall and the configuration of the overall IO-Link system can now be carried out from start to finish using a development tool.

This means that in a development tool of the applicant, the following steps are necessary when configuring an IO-Link system, with similar or comparable procedures being required by other development tools:

1) Configuration of the IO-Link master

All IO-Link masters available from the applicant are already contained in the hardware selection catalog. A specific IO-

Link master is selected e.g. by dragging & dropping the selected device into the hardware configuration.

The parameters of the IO-Link master can then be set in the development tool via a properties window of the object representing the IO-Link master.

2) Configuration of IO-Link devices
All IO-Link devices available from the applicant are already contained in the hardware selection catalog. Certified devices from third-party suppliers can be integrated into the database used by the development tool via their device description file, referred to in specialist terminology as an IODD. A specific IO-Link device is selected, for example by dragging & dropping the selected device into a configuration application.

The parameters of the IO-Link device can then be set in the development tool via a properties window of the object representing the IO-Link device.

3) Configuration of IO-Link subunits
IO-Link subunits (subdevices, feeders) can be selected in the hardware selection catalog. A graphic configuration is now possible by dragging & dropping the individual IO-Link subunits.

Modular IO-Link devices are configured in the device view of the hardware configuration. As the number and type of IO-Link devices and subunits used are known here, a start and a length of a data area containing I/O addresses can be determined automatically.

The parameters of each IO-Link subunit can then be set in the development tool via a properties window of the object representing the IO-Link subunit.

4) IO-Link diagnosis
The diagnostic information for all modules is read out and displayed. In the case of diagnostic information called for an IO-Link master, this is the group diagnosis of the master (the diagnostic information corresponds to the status of an LED on the IO-Link master) or diagnostic information regarding the IO-Link devices accessible to the IO-Link master.

5) Diagnosis of modular IO-Link devices
In modular IO-Link devices, the status of the device LEDs in each case corresponds to the relevant data which the diagnostic information comprises (group error, group warning, etc.). The diagnostic information and a status of the inputs/outputs of the individual subunits are visible.

An exemplary embodiment of the invention is explained in detail below with reference to the drawings. Items or elements corresponding to one another are labeled with the same reference characters in all the figures.

FIG 1 shows an automation system having a plurality of IO-Link devices,

FIG 2 shows an IO-Link device in an embodiment as a modular IO-Link device,

FIG 3 shows a representation of IO-Link devices by a development tool and

FIG 4 shows a flow chart illustrating a method for creating objects in the development tool for IO-Link devices.

FIG 1 shows schematically in highly simplified form an automation system 10 for controlling and/or monitoring an industrial technical process 12 which is not shown in detail. The automation system 10 comprises at least one automation device 14, e.g. a programmable logic controller. The latter comprises an IO-Link master 16 for connecting sensors and/or actuators via the communication standard known as IO-Link. IO-Link devices 18, 20, 22 are connected to the IO-Link master 16 in a per se known manner via point-to-point connections. At least one of the connected IO-Link devices 18, 20, 22 is a modular IO-Link device 20 which - as FIG 2 shows in schematically simplified form - comprises a plurality of IO-Link subunits. FIG 2 also shows that the IO-Link device 20 may comprise one or more IO-Link subunits 24, 26, 28, of which just one functions as a header module 24. The IO-Link device 20 has slots for accommodating the subunits 24, 26, 28, and a device-internal bus 32 runs in a rack 30 inside the IO-Link device 20 to each slot so that all the subunits which an IO-Link device 20 comprises are communicatively connected and can be accessed specifically by the header module.

The IO-Link master 16, the IO-Link devices 18, 20, 22 and the point-to-point connections provided for communicatively connecting these units together form the IO-Link system (FIG 1), in which the IO-Link master 16 functions as a superordinate unit. One of the subunits 24, 26, 28 of the modular IO-Link device 20 is selected as a header module 24 for communicating with the IO-Link master 16. Communication with the IO-Link master 16 takes place directly only with this

header module 24. All the subunits 24, 26, 28 which the modular IO-Link device 20 comprises can be accessed indirectly in the IO-Link system via this header module 24, i.e. starting from the header module 24 via the device-internal bus 32.

For the configuration, parameterization, diagnosis, etc. of IO-Link devices, a development tool 34 (FIG 1) available as software is used. This software may be executed on a programming device 36 (FIG 1) or similar which can be connected at least temporarily directly or indirectly, e.g. via the Internet, to the automation system 10. The programming device 36, e.g. a personal computer, has for this purpose in a per se known manner a memory 38 and a microprocessor-like processing unit (not shown). When the development tool 34 is loaded in the memory 38, said tool can be executed by the processing unit.

For this purpose FIG 3 shows in a schematically simplified form a possible representation of the IO-Link object model. It is shown that, when a modular IO-Link device 20 is created with the development tool 34, a first object 40 is created to represent the modular IO-Link device 20, a second object 42 is created to represent the IO-Link rack 30 comprising or accommodating in the modular IO-Link device 20 the subunits 24, 26, 28, a third object 44 is created to represent the subunit selected and functioning as an IO-Link header module 24 and at least one fourth object 46 is created for each further subunit 26, 28 of the modular IO-Link device 20.

It can also be seen from FIG 3 that the modular IO-Link device 20 is connected to the IO-Link system (FIG 1) only via the header module 24 outwardly representing the modular IO-Link device 20, i.e. the third object 44 generated for the IO-Link

header module 24. A fifth object '48 represents a point-to-point connection between the IO-Link master and the modular IO-Link device 20. The IO-Link master 16 is represented by a sixth object 50. The representation by the development tool 34 and the linking of the individual objects on which the representation is based cause the IO-Link header module 24 to be communicatively accessible in the IO-Link system, specifically by the IO-Link master 16. Of course, a complex IO-Link system may comprise a plurality of IO-Link devices 18, 20, 22 and also a plurality of modular IO-Link devices 20. Depending on the nature and scope of the IO-Link system, its representation by the development tool 34 will relate to a corresponding plurality of respective objects.

In a particular embodiment of the software tool 34, when an object 40 is created for a modular IO-Link device 20 the object 44 for the IO-Link header module 24 and/or the object 42 for the IO-Link rack 30 are created automatically. An object 40 is created for a modular IO-Link device 20, for example by the user of the software tool selecting the particular modular IO-Link device 20 in a hardware catalog and placing it in the automation solution using operator actions, e.g. drag & drop, that are now commonplace.

The software tool 34 in such an embodiment or in another embodiment is not shown separately, since this is to a certain extent only an additional or alternative software functionality of the software tool 34. A further embodiment of the software tool 34 provides that, when an object 42 for the IO-Link rack 30 is created automatically, objects 46 for the subunits 26, 28 which can be accommodated by the IO-Link rack 30 are created automatically. Furthermore, with respect to the functionality of the software tool 34 it is optionally

provided that, when objects 42, 44, 46 are created automatically, an interconnection is also created automatically between the automatically created objects 42, 44, 46. The interconnection created in this respect corresponds to the interconnection represented schematically in FIG 3 and provides access, for example starting from the object 44 representing the header module 24, to the object 42 representing the IO-Link rack 30 and indirectly to the objects 40, 46 representing the modular IO-Link device 20 and/or representing the subunits 26, 28 which the IO-Link rack 30 comprises or which the IO-Link rack 30 can accommodate.

FIG 4 makes this aspect, i.e. the functionality in this regard of the software tool 34, clear in a schematically simplified manner using a flow diagram: When objects (first function block 52) are created with the software tool 34, a check is made as to whether the object created or the object type which was selected for creating an object is an object 40 representing a modular IO-Link device 20. If this is the case, the tool branches to a second function block 54 by means of which object 42 for the IO-Link rack 30 is created automatically. If the software tool 34 is a software tool 34 in the particular embodiment already described above, a check is made in an optional fourth function block 56 as to what type of IO-Link rack IO-Link rack 30, for which object 42 was created as a proxy, is, and then (fifth function block 58) objects 46 are created automatically for subunits 26, 28 which the IO-Link rack 30 comprises or which the IO-Link rack 30 can accommodate. If the physical IO-Link rack 30 represented by the object 42 does not yet have any subunits 26, 28 plugged in it, placeholder objects are generated as objects 46; if the IO-Link rack 30 already has subunits 26, 28 in individual or in all its slots, the objects generated automatically can be

generated in respect of the subunits 26, 28 which are actually plugged in, by importing for the latter e.g. data from the relevant device description.

Claims

1. A method for operating an automation system (10), wherein the automation system (10) comprises, in communicatively connected form, a superordinate unit (16), which is an IO-Link master, and at least one modular IO-Link device (20), wherein the modular IO-Link device (20) has a device-internal bus (32) and subunits (24, 26, 28) which can be addressed by means of the latter and which the modular IO-Link device (20) comprises, wherein communication with the modular IO-Link device (20) involves the selection of one of the subunits (24, 26, 28) thereof as a header module (24), and the communication takes place only with the header module (24) directly and via the header module (24) with the other subunits (26, 28) of the modular IO-Link device (20) indirectly, wherein the IO-Link device (20) has slots to accommodate the subunits (24, 26, 28) and inside the IO-Link device (20) the device-internal bus (32) runs in a rack (30) to each slot such that all the subunits (24, 26, 28) which the IO-Link device (20) comprises can be accessed by the header module (24).

2. The method for operating an automation system (10) as claimed in claim 1 or a method for configuring or parameterizing a modular IO-Link device (20) in an automation system (10), wherein, when a modular IO-Link device (20) is created in a development tool (34), a first object (40) is created to represent the modular IO-Link device (20), a second object (42) is created to represent an IO-Link rack (30) in the modular IO-Link device (20), said rack comprising or accommodating the subunits (24, 26, 28), a third object (44) is created to represent the subunit that is selected and functions as an IO-Link header module (24) and at least one

fourth object (46) is created for each further subunit (26, 28) of the modular IO-Link device (20).

3. The method as claimed in claim 2, wherein, when an object (40) is created for a modular IO-Link device (20), an object (44) is automatically created for the selected subunit of the modular IO-Link device (20), said subunit functioning as the IO-Link header module (24).

4. The method as claimed in claim 3, wherein, when an object (40) is created for a modular IO-Link device (20), an object (42) is automatically created for the IO-Link rack (30) of the modular IO-Link device (20).

5. The method as claimed in claim 4, wherein, when an object (42) is automatically created for the IO-Link rack (30), objects (46) are automatically created for subunits (26, 28) that can be accommodated by the IO-Link rack (30).

6. The method as claimed in claim 3, 4 or 5, wherein for the or each automatically created object (42, 44, 46) the automatically created object (42, 44, 46) is automatically interconnected with the object (40) to represent the modular IO-Link device (20) and/or with other automatically created objects (42, 44, 46).

7. The method as claimed in claim 1, wherein the IO-Link device comprises three subunits.

8. A computer program having program code means for implementing the method as claimed in any one of claims 1 to 7 when the computer program is executed on a computer, in particular on a programming device for creating and editing a

computer program as an automation solution for controlling and/or monitoring a technical process.

9. A computer program product comprising a computer program as claimed in claim 8 that can be executed by a computer.

10. A programming device (36) for creating and editing a computer program as an automation solution for controlling and/or monitoring a technical process, said programming device having a memory (38) into which a computer program as claimed in claim 8 is loaded as a development tool (34).

FIG 1

(Prior art)

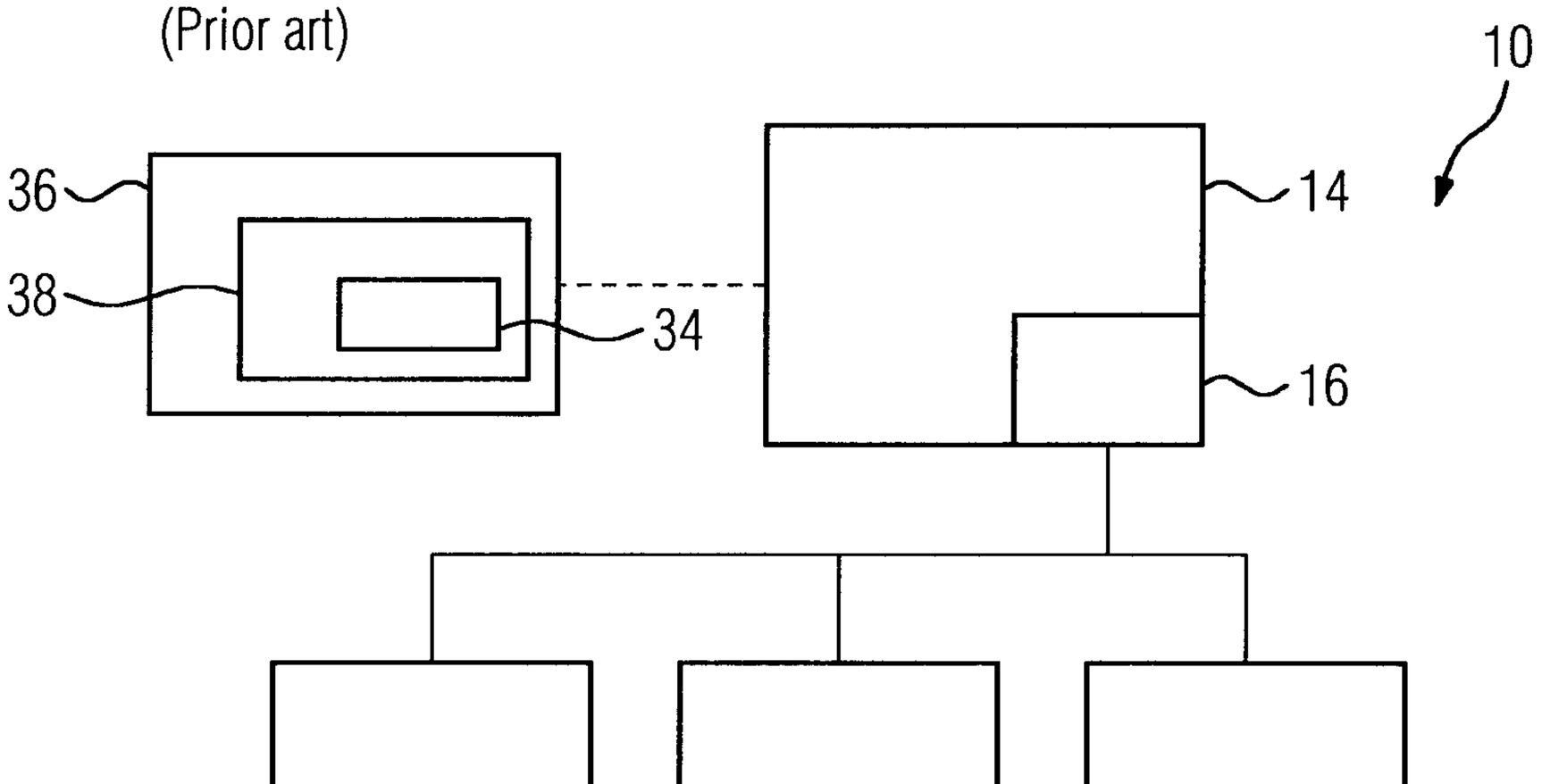


FIG 2

(Prior art)

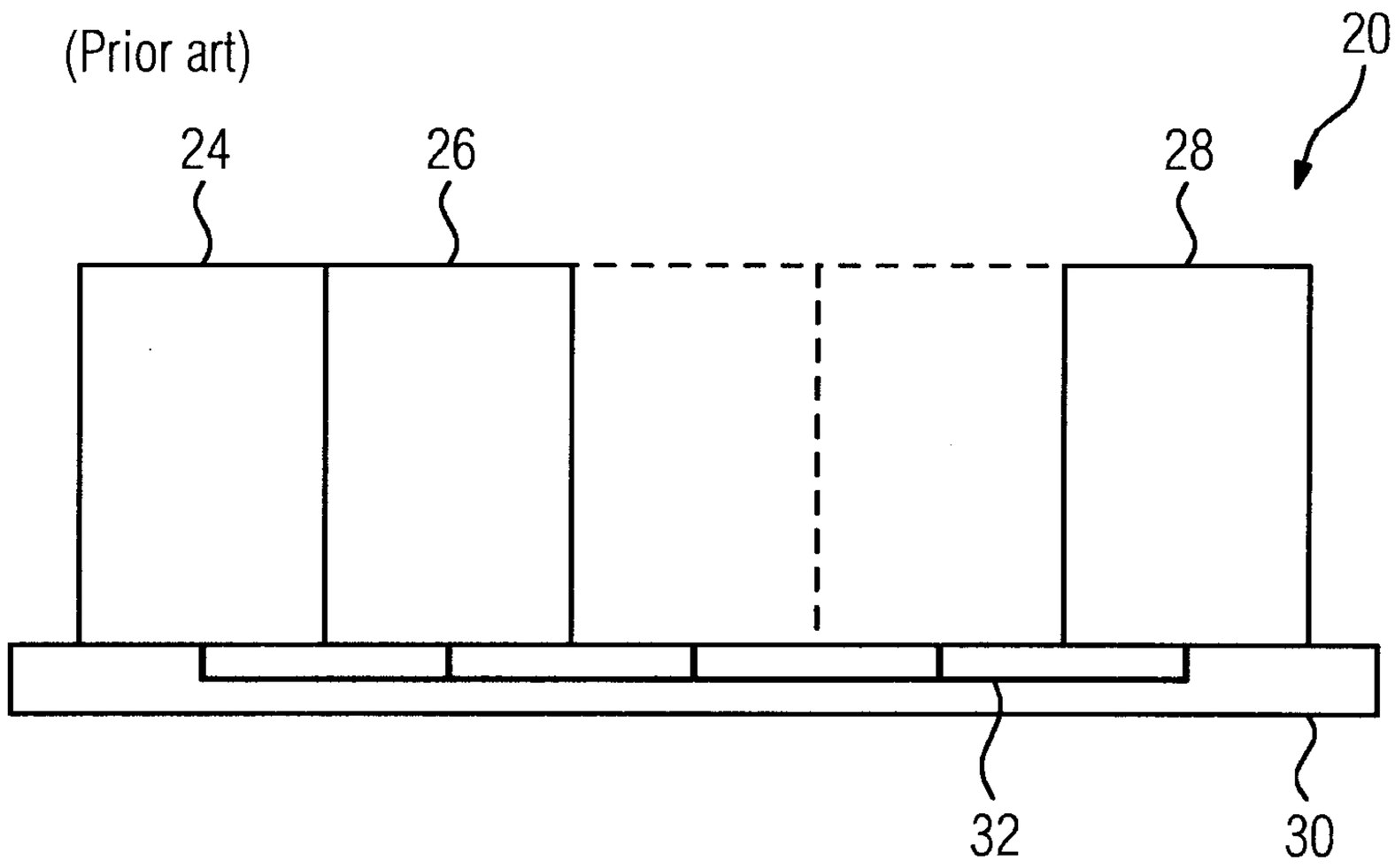


FIG 3

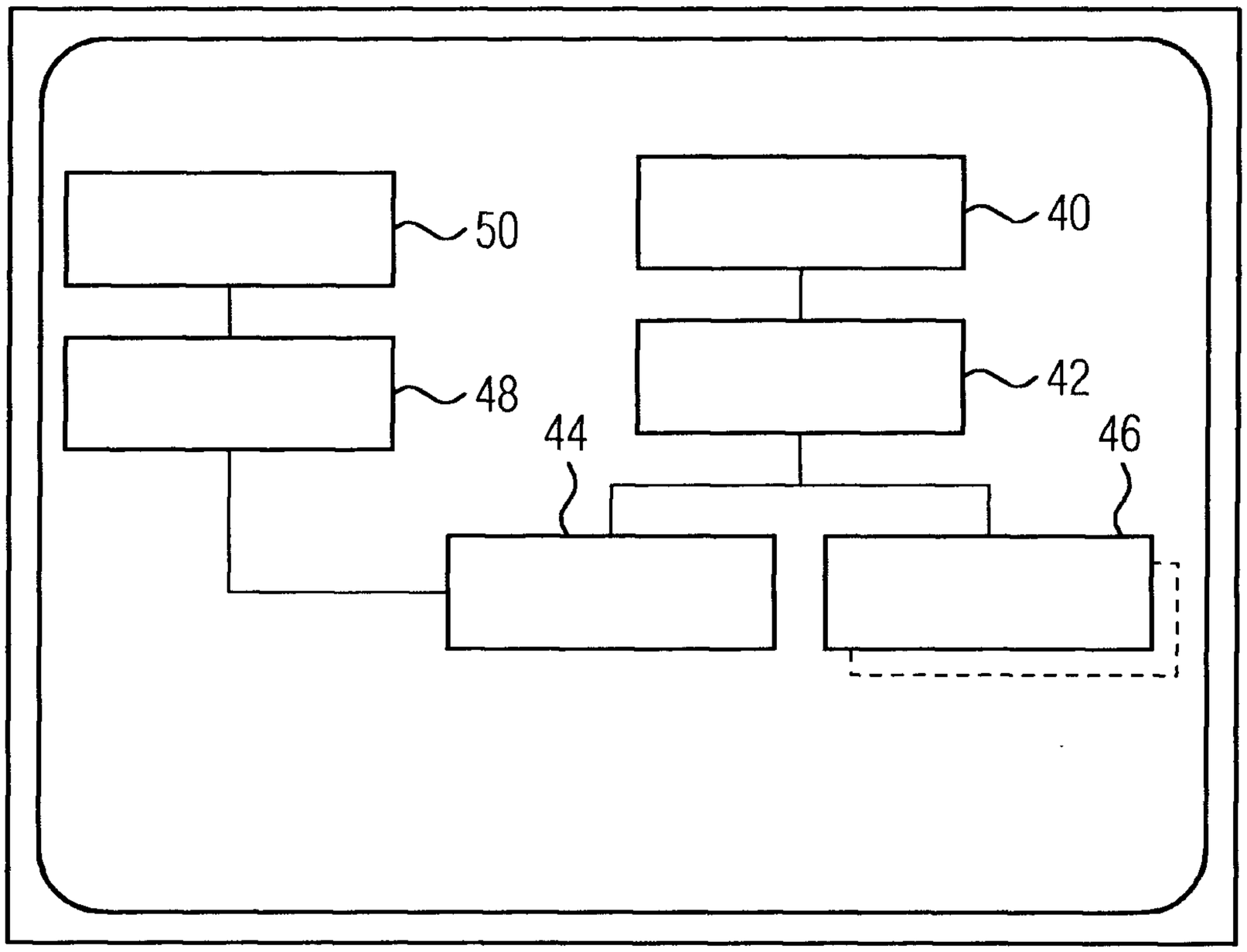


FIG 4

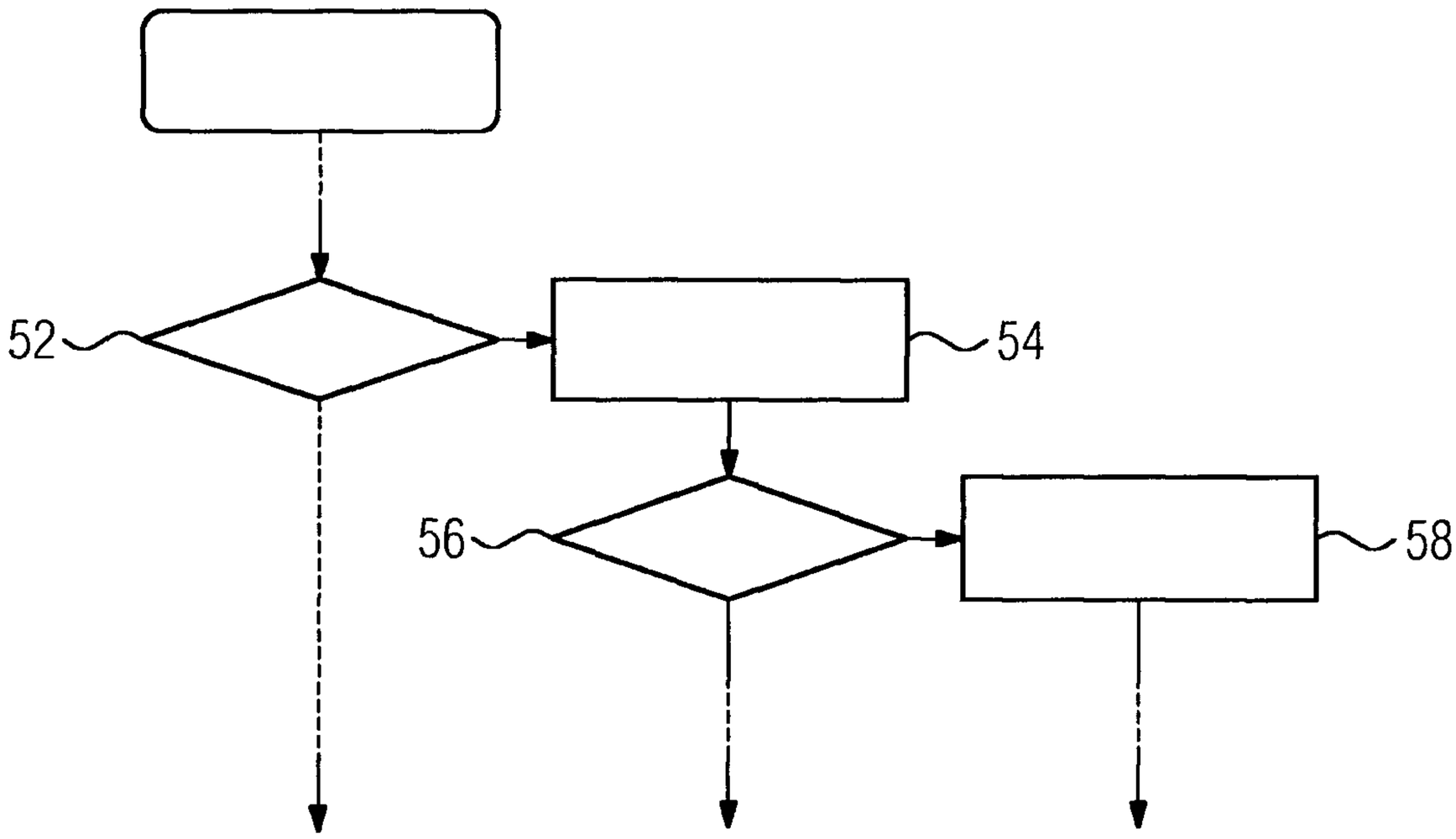


FIG 3