(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
02 April 2020 (02.04.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/068095 A1

(51) International Patent Classification:
H04L 12/24 (2006.01) *H04L 12/403* (2006.01)

(21) International Application Number:
PCT/US2018/053287

(22) International Filing Date:
28 September 2018 (28.09.2018)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 10300 Energy Drive, Spring, Texas 77389 (US).

(72) Inventors: **DANG, Thieu X.**; Columbia Tech Center, 1115 SE 164th Ave., Vancouver, Washington 98683 (US). **SORIANO FOSAS, David**; Columbia Tech Center, 1115 SE 164th Ave., Vancouver, Washington 98683 (US). **BALDO, Nicola**; Cami de Can Graells, 1-21, 08174 Sant Cugat del Valles (ES).

(74) Agent: **LEMMON, Marcus** et al.; HP Inc., 3390 E. Harmony Road, Mail Stop 35, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:
— *as to the identity of the inventor (Rule 4.17(i))*

(54) Title: MASTER/SLAVE COMMUNICATION

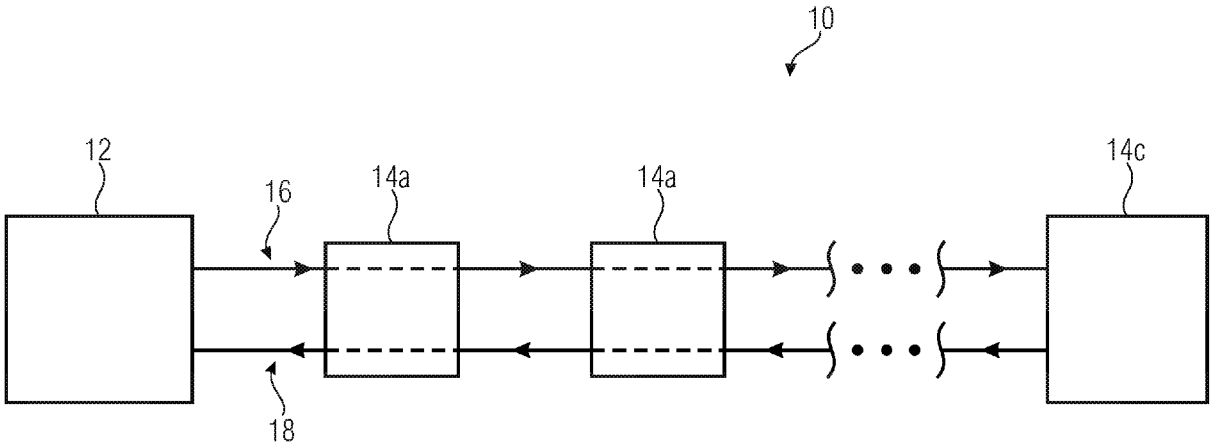

Fig. 1

(57) **Abstract:** A communication system comprises a master node and slave nodes, wherein the master node and the slave nodes are coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node. Each node comprises a hardware interface to receive an incoming frame. The hardware interface detects an end of the incoming frame using a break character. Each node checks whether the incoming frame has a correct frame length, further processes the incoming frame if the incoming frame has the correct frame length, and does not further process the incoming frame if the incoming frame does not have the correct frame length.

WO 2020/068095 A1

[Continued on next page]

Published:
— *with international search report (Art. 21(3))*

MASTER/SLAVE COMMUNICATION

BACKGROUND

[0001]Appliances, such as 3D printers and large format printers, may include large numbers of sensors and actuators of many different kinds, such as temperature, pressure, humidity and presence sensors, encoders, motor drivers, fans, heaters, etc. A controller, such as a print engine, is to interface with all these devices in order to control processes, such as printing processes, appropriately.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Examples will now be described, by way of non-limiting examples, with reference to the accompanying drawings, in which:

[0003] Fig. 1 is a block diagram of a communication system according to an example;

[0004] Fig. 2 is a more detailed block diagram of a communication system according to an example;

[0005] Fig. 3 is a schematic diagram showing master node and a slave mode according to an example;

[0006] Fig. 4A and 4B show examples of message formats for a read request and a response to the read request;

[0007] Fig. 5A and 5B show examples of message formats for a write request and a response to the write request;

[0008] Figs. 6 is a time diagram showing a break character according to an example;

[0009] Fig. 7 is a schematic view of an example of a node interface;

[00010] Fig. 8 is a flow chart of an example of a method of communicating in a communication system;

[00011] Fig. 9 is a flow chart of an example of features of a master/slave communication at a slave node side; and

[00012] Fig. 10 is a flow chart of an example of features of a master/slave communication at a master node side.

DETAILED DESCRIPTION

[00013] Generally, examples of the present disclosure relate to master/slave communication and aspects of master/slave communication between a master node and slave nodes in a communication system using a master/slave communication protocol, MSCP. Examples relate to a communication system designed for networks of sensors and actuators that is robust to communication errors and machine readable instruction errors. In examples, the communication system is targeted at large networks of low cost devices provided with limited computational resources. The error resilience and robustness characteristics of the communication system are designed to maintain a good trade-off with the communication efficiency of the master/slave communication protocol used.

[00014] Generally, examples of the present disclosure may relate to a master/slave communication protocol for realizing a low-cost network of slave devices, such as sensor/actuator devices, to be connected to a master device,

such as a central engine in a 3D/large format printer. Other examples may be directed to other products in which communication between a number of slave devices and one master device is desired. Examples of the present disclosure provide a low-latency, high bandwidth connectivity in such system and may scale up to very large numbers of devices. In examples, devices for implementing the protocol may be realized using price competitive embedded microcontrollers, and hence may provide a very cost-effective solution.

[00015] Examples of the present disclosure are designed for a complex appliance or product, such as a 3D printer, which is composed of a variety of sensor and actuator devices which are controlled by a central engine. For the appliance to work properly, the interaction between the central engine and the sensors and actuators has to be successful and timely, so that the engine may perform its internal routines using up-to-date sensorial input and perform the intended actuation with the proper timing.

[00016] Several causes of errors may possibly hinder the interaction of the central engine, the master node, with sensor and actuator devices, the slave nodes. Such causes may be communication errors due to noise and interference, malfunctions of the slave nodes, e.g., due to a machine readable instruction error, or due to unexpected latencies in the device readable instructions. A further cause may be a malfunction of the central engine, e.g., a hang in the central engine when executing machine readably instructions.

[00017] What is desired is an error resilient communication system that may satisfactorily address such causes of error, and that may be implemented on low cost devices with limited computational capabilities, but also supporting high communication bandwidth. In examples, an electrical communication interface using universal asynchronous serial communication, UART, is used to implement the MSCP. In examples, the MSCP provides a resilient and efficient communication protocol that may work on top of a UART electrical interface and tackle the aforementioned types of errors.

[00018] Examples of the present disclosure may be implemented using a master/slave communication protocol, MSCP. According to the MSCP, a host, master node, communicates with the devices, slave nodes, with read requests and write requests to specific registers, which provide the interface for the master node to interact with the slave nodes, which may comprise sensors and actuators. The MSCP may be transported over a full duplex UART physical interface. Before specific features making the MSCP error resilient are described, features of the MSCP are described.

[00019] Examples of the present disclosure provide a communication system in which the MSCP is implemented. Fig. 1 shows a schematic view of a communication system 10 comprising a master node 12 and slave nodes 14a, 14b, and 14c. The master node 12 and the slave nodes 14a, 14b, 14c are coupled to each other in a daisy chain configuration. The daisy chain configuration includes a request line 16 to transmit requests from the master node 12 to the slave nodes 14a, 14b, and 14c, and a response line 18 to transmit responses from the slave nodes 14a, 14b, and 14c to the master node. Generally, as indicated by the interruption and the dots in Fig. 1, the communication system 10 may comprise a different number of slave devices, such as a larger number or a lower number when compared to the three slave devices shown in Fig. 1.

[00020] The term daisy chain configuration is meant to define a configuration, in which a first slave node, i.e., slave node 14a, is coupled to the master node 12 via the request line 16 and the response line 18, and in which each subsequent slave node is coupled to the respective preceding slave node via the request line 16 and the response line 18. Thus, the slave nodes are serially coupled to each other to form a chain. The request line 16 may be regarded as being formed by all request line portions coupling the master node and the slave nodes of the daisy chain configuration to each other and the response line 18 may be regarded as being formed by all response line portions coupling the master node and the slave nodes of the daisy chain configuration to each other.

In the daisy chain configuration, each node transmits data to the next node in the chain if appropriate. For example, master node 12 sends a request in which node 14b is addressed via request line 16 to node 14a, and node 14a forwards the request to node 14b since node 14a is not addressed in the request. A response from node 14b to the master node 12 will then be sent from node 14b to node 14a and forwarded to master node 12 by node 14a. Request line 16 and response line 18 represent unidirectional data lines, wherein the respective direction of data communication is indicated by arrows in Fig. 1.

[00021] Each of the master node and the slave nodes may be implemented in hardware using discrete modules and/or data processing components that are not limited to any particular hardware and machine-readable instruction configuration. The nodes may be implemented using analogue and/or digital hardware components, such as application specific integrated circuits, field programmable gate arrays, digital signal processors, microprocessors and microcontrollers. The nodes may be implemented in any computing or data processing environment including hardware components, such as processors and memory devices, and machine-readable instructions. The machine-readable instructions may be stored in any appropriate memory and may be executed by the processor in order to achieve the functionalities and processes described herein. In some implementations, the functionalities are combined into a single data processing component. In other implementations, the respective functionalities may be performed by a respective set of multiple data processing components. The memory devices may store process instructions, machine-readable instructions, for providing the functionality and implementing the methods described herein. The memory devices may include tangible machine-readable storage media. Memory devices suitable for embodying these instructions and data include all forms of computer-readable memory, including, for example, semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices, magnetic disks such as internal hard disks and removable hard disks, magneto-optical disks, and ROM/RAM devices. Accordingly, in examples, the nodes may be implemented in hardware

or in a combination of hardware and machine readable instructions to implement the communication protocol described herein.

[00022] The terms master and slave are used herein to indicate that the master node is allowed to initiate communication and to transmit requests via the request line, while the slave nodes are not allowed to transmit responses via the response line without having received an associated request from the master.

[00023] Fig. 2 shows a more detailed schematic view of a communication system, wherein a master node 12 and two slave nodes 14a and 14b are shown. Again, a different number of slave nodes may be provided as indicated at 20 in Fig. 3. The master node may comprise a master printed circuit assembly, PCA, 22 and a master integrated circuit, IC, 24. A transmitting output TX of the master IC is coupled to a request line MOSI, Master Out Slave In, which represents an example of a request line 16. A receiving input RX of the master node is coupled to a response line MISO, Master In Slave Out, which represents an example of a response line. The direction of data transmission over the MOSI line and the MISO line are indicated in Fig. 2 by respective arrows. Master node 12 may further include a transmission driver 26 and a receiving driver 28.

[00024] Each slave node may include a slave IC 30. A receiving input RX of each slave node 14a, 14b is coupled to the request line MOSI and a transmission input TX of each slave node 14a, 14b is coupled to the response line MISO. In the example shown, in each of slave nodes 14a, 14b, the request line MOSI is further coupled to a first input of an OR gate 32 and a disable output of each slave node 14a, 14b is coupled to the second input of the OR gate 32. The output of the OR gate 32 is coupled to the MOSI line connecting the corresponding slave node to the next slave node. This allows the slave node to disable the request line MOSI to not forward the request to the next slave node using some previous request or condition. In other examples, such a

functionality may be implemented differently. In the example shown, the transmission output RX of slave IC 30 is coupled to a first input of an AND gate 34 and the MISO line coming from a preceding slave node is coupled to the second input of the AND gate 34 and the output of the AND gate 34 is coupled to the MISO line connecting the respective slave node to the next slave node.

[00025] In the above example, idle levels of the MOSI line and the MISO line are low and OR gate 32 and AND gate 34 are used. In other examples, idle levels of the MOSI line and the MISO line may be high. In such examples, OR gate 32 may be replaced by an AND gate and AND gate 34 may be replaced by an OR gate.

[00026] In examples, a slave node N may raise or lower the disableNext signal after successful reception of a write register request addressed to the device N with a certain register address and value. Depending on the value of the disableNext signal subsequent transmissions by the master will be propagated by device N to device N+1 or will be blocked by device N, and, therefore, not propagated to device N+1. In examples, the OR gates 32 may be omitted.

[00027] As indicated in Fig. 2, a configuration line CONFIG may be provided between the master node 12 and the first slave node 14a and between respective adjacent slave nodes. The configuration line may be used to communicate between the master node 12 and the slave nodes 14a, 14b control information useful to implement the communication protocol described herein.

[00028] In operation, the master node 12 will send requests via line MOSI, each request addressed to a specific one of the slave nodes. Each slave node may have associated therewith a specific device identity, ID, and, therefore, may be addressed via a device ID in the request. In examples, the addressed slave node will not forward the request to the next slave node in the chain but will transmit a response to the master node via line MISO. Transmission of

requests and responses will take place according to the MSCP described herein.

[00029] In examples, the master/slave communication protocol, MSCP, may be described using a layer structure as shown in Fig. 3. It may comprise a physical layer specification 40, on top of a universal asynchronous receiver transmitter, UART, baseline 42. Thus, examples may enjoy hardware peripheral availability across a broadest set of microcontrollers possible, especially including the cost effective ones. Thus, in examples, each of the master node and the slave nodes comprises a UART interface, wherein a transmit output of the UART interface of the master node is coupled to the request line, a receive input of the UART interface of the master node is coupled to the receive line, transmit outputs of the slave nodes are coupled to the receive line, and receive inputs of the slave nodes are coupled to the transmit line.

[00030] In examples, the MSCP further comprises a medium access control, MAC, layer specification using a master-slave paradigm, MAC/LC layer 44 in Fig. 3, wherein LC stands for link control. In examples, this specification focuses on two types of request-response frame exchanges. In examples, the MSCP supports two request types, read requests and write requests, resulting in corresponding read and write operations at the addressed slave node. Thus, in examples, sensor and actuator use cases may be supported efficiently, while at the same time the design of the protocol may be kept simple and well-suited for hardware acceleration on the master side. The medium access control specification aims at providing low latency while ensuring collision-free operation when multiple devices are sharing the same physical bus. In addition, an MSCP Application Protocol Layer 46 may provide support for a device control protocol specification that defines procedures for device identification, initialization and remote upgrade. Such support may take place via a configuration line config and corresponding interfaces in the master node 12 and the slave node 14. The MSCP may further provide support for application protocols aimed at supporting application-specific sensors and actuators, whose

interaction with the central engine is in examples of the present disclosure the purpose of the MSCP protocol. Corresponding application layers 48 are shown in Fig. 3. On the master side, application layer 48 comprises a MSCP master control application and master specific-purpose applications, and on the slave side, application layer 48 comprises a MSCP slave control application and slave specific-purpose applications.

[00031] On the master side, the protocol layer stack architecture may further comprise a general-purpose input/output, GPIO. In addition, the protocol layer stack architecture may further comprise a UART bootloader on the slave side and a UART bootloader client on the master side, such as a STM32® UART bootloader and a STM32® bootloader client.

[00032] In examples, the master/slave communication protocol may be implemented in the form of a set of machine readable instructions components for the master to communicate with the slave devices. In examples, the master node may be a central engine of a printer and the slave nodes may provide access to sensors and actuators of the printer. Generally, the master node may be a controller of an appliance and the slave nodes may be components of the appliance that provide information and/or components of the appliance that are controlled by the controller. In examples, the master/slave communication protocol may be implemented in the form of a device machine readable instructions library to develop application-specific MSCP devices to be tailored to the features of different products. In examples, the master/slave communication protocol may be implemented in the form of several application-specific MSCP device realizations.

[00033] In examples, the data communication path for implementing the MSCP comprises two physical lines, the MOSI line representing a request line, and the MISO line representing a response line. On the master or host side, the MOSI line may be connected to an UART TX output and the MISO line may be

connected to the UART RX input. On each slave device, the MOSI line may be connected to the UART RX input and the MISO line to the UART TX output.

[00034] Interactions according to the MSCP are of the type master/slave, with the master sending a request on the MOSI line to a specific slave device and the intended slave device sending the response on the MISO line to the master.

[00035] In examples, just two transactions types are defined by the protocol, a read register request/response and a write register request/response. The size, i.e., frame length or length, of each message is fixed and asymmetric between the request and the response. In examples, the protocol prescribes that the requests are either a write register request having a fixed first frame length or a read register request having a fixed second frame length, and that a response to a write register request is a write register response having a fixed third frame length and that a response to a read register request is a read register response having a fixed fourth frame length. In examples, the first frame length is larger than the second and third frame lengths and the fourth frame length is larger than the second and third frame lengths. In examples, the first and fourth frame lengths are the same and the second and forth frame lengths are the same. In examples, for read register transactions, the frame length of the request is 3 bytes and the frame length of the response is 7 bytes. In examples, for write register transactions, the frame length of request is 7 bytes and the length of the response is 3 bytes. In examples, the frame lengths may be adjusted as described above in order to maximize protocol throughput while maintaining protocol simplicity.

[00036] In examples, the MSCP prescribes that request messages, which are also simply called requests, have a specific format. The requests transmitted according to the MSCP contain a request protocol header. The request protocol header contains a field defining the intended receiver device, i.e., slave node. In examples, the request protocol header may include the device identity, ID, of the slave node addressed by the request. In addition, the request protocol

header may include a register address for the requested transaction. In examples, different address ranges may be defined for MSCP control registers and for application registers, and the request may include a register type indicator to indicate whether a MSCP control register or an application register is to be accessed. In addition, the request includes a request type indicator to indicate the type of the request. In examples, the requested transaction is a read transaction or a write transaction. In examples, no other transaction types are possible. The register type indicator and/or the request type indicator may be included in the request header. The requests may further include data fields. In examples, a request does not include data fields if the request is a read request and a request includes data fields, such as data fields of a length of four bytes, if the request is a write request. In examples, the requests may further include a checksum, such as a checksum of 8 bits length. In examples, each request is followed by a break character, which may be a formed by a number of consecutive bits of the same value, such as a number of consecutive zero bits.

[00037] In examples, the MSCP prescribes that response messages, which are also simply called responses, have a specific format. The responses include a response protocol header. The response protocol header contains a field indicating the device ID of the slave device transmitting the response. In addition, the header may include a status field containing status information on the slave device transmitting the response. In case of a response to a read request, the response includes data fields, such as data fields of a length of 4 bytes. The responses may include a checksum, such as a checksum of 8 bits length. In examples, each response is followed by a break character, which may be formed by a number of consecutive bits of the same value, such as a number of consecutive zero bits. The responses may include frame bits separating further information

[00038] Examples of a request and a response in case of a read transaction are shown in Figures 4A and 4B and examples of a request and a response in case of a write transaction are shown in Figures 5A and 5B. The respective

headers have a length of two bytes. As shown, frame bits may separate device ID and address fields in the protocol headers of the requests, and frame bits may separate device ID and status fields in the protocol headers of the responses. As shown in Figures 4A and 4B, a request type indicator comprises a bit RnW. If RnW has a first value, such as one, the request is a read request and if RnW has a different second value, such as zero, the request is a write request. A register type indicator comprises a bit UnR. If UnR has a first value, such as one, the MSCP control register is to be accessed and if UnR has a different second value, such as zero, the application register is to be accessed. The requests and responses comprise a checksum CRC at the end thereof. Each message is followed by a break character BREAK. In the responses, an attention bit ATTN may be inserted preceding the checksum. According to Fig. 4A, the overall length of a read request plus the break character may be four characters. According to Fig. 4B, the overall length of a response to the read request plus the break character may be eight characters. According to Fig. 5A, the overall length of a write request plus the break character may be eight characters. According to Fig. 5B, the overall length of a response to the write request plus the break character may be four characters. Each character may include 1 byte.

[00039] The MSCP control registers may include control data used in implementing the MSCP. In examples, the slave nodes comprise sensor nodes and actuator nodes. In such examples, each type of slave node, i.e., receiver device, is expected to support a well-defined set of application registers, mapped to the sensors and actuators that are supported by the particular device type. By sending appropriate read register requests and write register requests, the master node, such as a print engine, may retrieve sensor data and apply control actions on actuators by communicating with the slave nodes via the master/slave communication protocol described herein.

[00040] In examples, the MSCP disclosed herein is designed for providing low latency communications. In examples, the allowed latencies may be as small as

from tens to a few hundreds of microseconds and are enforceable on a per slave device basis. In examples, the MSCP may be implemented with high speeds, such as up to 6Mbps or up to 9 Mbps, wherein Mbps stands for megabit per second. Higher protocol speeds are possible depending on cable length and signal integrity, wherein cable shielding and proper impedance construction may help increasing the maximum protocol speed.

[00041] In examples, the timing of requests/responses is accurately controlled in order to maximize the system throughput while at the same time avoiding collisions on the response line. In examples, the master node determines the timing of a response sent by a slave node in response to a request. To be more specific, the protocol described herein prescribes that a response type and size are univocally associated with each request type. Thus, when the master node sends a request to a slave node, the associated response type and size, and hence duration, are univocally determined by the request type. The corresponding associations between request types, response types and sizes may be stored at each node, such as in respective MSCP control registers.

[00042] In examples, a slave node is allowed to transmit a response on the response line if it is able to decode the request correctly. A slave node is the addressed slave node if the device ID in the request corresponds to the device ID of the slave node. The addressed slave node then verifies the checksum and starts transmission of the response if the checksum is verified successfully. The slave node does not start transmission of the response if the checksum is not verified successfully. No other device beside the slave node addressed in the request is allowed to transmit a response.

[00043] In examples, the MSCP may prescribe that the transmission of a response on the response line has to be finished after a fixed duration counting from the end of the request on the request line, which resulted in the response. To this end, the slave node is to start transmission of the response to the current request within a predetermined time window from the end of the current request. The fixed duration after which the response has to be finished is

determined to allow enough time to transmit the response of the fixed length determined by the request type plus an adequate margin to account for worst case signal propagation and processing time tolerances at the slave node side. Thus, the master node may expect receiving the response within a specific time window. The master node will defer start of a next transmission on the request line for a time that is sufficient for the end of the next transmission to occur after the end of the response to the current request. Thus, collision of responses on the response line may be prevented.

[00044] Examples of the present disclosure may be implemented using the MSCP protocol described and may comprise some or all features thereof. Examples of the present disclosure are directed to features making a master/slave communication error resilient.

[00045] Examples of the present disclosure provide a communication system comprising a master node and slave nodes, wherein the master node and the slave nodes are coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node. Each node comprises a hardware interface to receive an incoming frame, wherein the hardware interface detects an end of the incoming frame using a break character. Each node checks whether the incoming frame has a correct frame length, to further process the incoming frame if the incoming frame has the correct frame length, and not to further process the incoming frame if the incoming frame does not have the correct frame length.

[00046] Examples of the present disclosure provide a slave node for a communication system comprising a master node and slave nodes coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node. The slave node comprises a hardware interface to receive an

incoming request frame, wherein the hardware interface detects an end of the incoming frame using a break character. The slave node checks whether the incoming request frame has a correct frame length, further processes the incoming frame if the incoming frame has the correct frame length, and does not further process the incoming frame if the request frame does not have the correct frame length.

[00047] Examples of the present disclosure provide a master node for a communication system comprising the master node and a plurality of slave nodes coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node. The master node comprises a hardware interface to receive an incoming response frame. The hardware interface detects an end of the incoming frame using a break character. The master node checks whether an incoming response frame has a correct frame length, further processes the incoming response frame if the incoming response frame has the correct frame length, and does not further process the incoming response frame if the response frame does not have the correct frame length.

[00048] The correct frame length is the fixed frame length associated with a respective response or request. At the slave side, the correct frame length may include frame lengths of both, a write request and a read request, while, at the master side, the correct frame length may include a single frame length, namely the frame length of the response type associated with the sent request.

[00049] Examples of the present disclosure use frames having a fixed frame length and further processing of a frame depends on whether an incoming frame has the correct fixed frame length or not. If the frame dos not have the correct frame length it is assumed that there is a form error and, therefore, the frame is not further processed. Thus, superfluous processing and the workload associated therewith may be avoided. The frame length may be determined in an easy manner using the break character. In examples, the break character

may be formed by a number of at least ten bits having the same level different from an idle level of the request line and the response line so that it may be detected easily. In examples, if the idle level is high, the break character has a number of at least ten low bits.

[00050] In examples, further processing an incoming frame includes validating a checksum of the frame and any processing following the validation, such as sending the frame to a link layer for further processing. Thus, working load associated with validating the checksum and with any further processing may be avoided if the incoming frame does not have the correct frame length.

[00051] In examples, further processing the incoming frame by a slave node comprises sending a response to the master node if the incoming frame is addressed to that slave node, and forwarding the incoming frame to the next stage of the daisy chain if the incoming frame is not addressed to that slave node. This further processing is not performed if an incoming frame has the wrong frame length. Thus, superfluous working load may be avoided in an addressed slave node and in all slave nodes in the daisy chain downstream of a slave node detecting a wrong frame length.

[00052] In examples, there are just two request types and two response types, wherein each request type and each response type has an associated frame length. Thus, detection whether an incoming frame has been received properly may be done in an easy manner by detecting whether the incoming frame has the correct frame length. In examples, the request types are a read request and a write request and the response types are a read response and a write response. In examples, the frame length of a write request is larger than the frame length of a read request, and the frame length of a read response is larger than the frame length of a write response. A read request may have a first fixed frame length, such as three bytes, a write request may have a second fixed frame length, such as 7 bytes, a read response may have a third fixed frame length, such as 7 bytes, and a write response may have a fourth fixed frame length, such as 3 bytes. Thus, the frame lengths may be adjusted

according to the type of the frame so that the requests and responses may be kept as short as possible.

[00053] In examples, the hardware interface is to transfer the incoming frame to a memory using direct memory access, DMA. This permits the incoming frame to be received in a fail-safe manner.

[00054] In examples, the master node is to retransmit a request frame once or N times and wherein the master node is to increase a count in a header of this request frame every time the request frame is retransmitted. In examples, the master node is to retransmit the request frame upon expiration of a timeout since transmitting the request frame without receiving a response on the response line, and/or if the incoming frame does not have the fixed frame length, and/or if a checksum in the incoming frame is not validated successfully. Thus, recovery from errors may be achieved in case an incoming frame is not received properly by the master node or one of the slave nodes. If an incoming frame is not received by an addressed slave node properly, the slave node will not transmit a response in time and this will be recognized by the master node which is then retransmitting the frame.

[00055] Since the master increases a count in a header of the request frame every time the request frame is retransmitted, a slave node which received a request frame successfully may discard any retransmission of the request frame having a higher count. Thus, superfluous responses to such a request frame may be avoided.

[00056] In examples, each slave node comprises a hardware timer and the slave node addressed in a request transmits a response within a predetermined time window using the hardware timer. Thus, transmitting the response in time if the request frame is received properly may be ensured.

[00057] Examples of the present disclosure provide an appliance, such as a printer, 3D printer or larger format printer, comprising a communication system of the present disclosure, wherein the master node comprises a central

controller of the appliance and the slave nodes comprise sensors and actuators of the appliance. In examples, the sensors and actuators may be of many different kinds, such as temperature, pressure, humidity and presence sensors, encoders, motor drivers, fans, and heaters. In such examples, the slave nodes are formed by sensors and actuators and the master node is formed by a controller, such as a printer controller.

[00058] In examples, the protocol, such as the MSCP described herein, specifies that each frame transmission is terminated by a break character. The break character comprises a predetermined bit sequence detectable by the node receiving the frame. In examples, the break character is a sequence of at least 10 low bits. Fig. 6 shows a schematic view of an end of a frame, i.e., two last characters 100 of a frame, followed by a break character 102. Each character comprises a start bit, eight information bits, and a stop bit. The break character 102 comprises at least 10 low bits, i.e., zero bits. Since the idle level of UART is high, the break character is the most robust signal that may be sent over UART because of its low-frequency nature. Additionally, the break character may automatically be detected by UART hardware peripherals and may cause a restart in the UART physical layer synchronization, i.e., the process of looking for the start bit of a new frame.

[00059] Hence, using of such a break character permits starting reception of a new frame from a clean synchronization state. The detection of the end of the frame may be robust as it is using a robust symbol. The detection of the end of the frame may be hardware driven, leveraging the functionality of UART HW peripherals, and thus not exposed to logical or timing errors in non-hardware driven approaches. In case of a hardware driven detection of the end of the frame, direct memory access, DMA, may be easily employed for the frame reception process without non-hardware driven reception which due to its stringent timing requirements is error prone.

[00060] Fig. 7 shows a schematic view of components of the master node 12 and the slave nodes 14, which are involved in break character detection and

direct memory access. Each node comprises a hardware interface 110. Hardware interface 110 comprises a receiving input RX, a break character detection circuit 112 and a reception buffer 114. The reception buffer is coupled to a memory 118 of a microcontroller 116 for DMA. An input of break character detection circuit 112 is coupled to input RX and an output of break character detection circuit 112 is coupled to microcontroller 116. The communication protocol, such as the MSCP described above, may be implemented using microcontroller 116. Hardware interface 110 may be implemented by a UART hardware peripheral and reception buffer 118 may be a one-byte reception buffer of the UART hardware peripheral.

[00061] An incoming frame is received at input RX. The break character detection circuit 112 is a hardware circuit and triggers a hardware signal at its output when detecting a break character to notify microcontroller 116 when a break character is seen on a line coupled to input RX. One possible implementation to realize this trigger is to implement the output of the break character detection circuit 112 as a binary output. Circuit 112 switches the binary output to a high level whenever the input thereof stays low for at least ten consecutive low bits and switches the output back to a low level when the input thereof switches to a high level.

[00062] Since each frame is terminated with a break character the system may be implemented with low computational complexity on microcontroller 116 that has hardware support for break character detection and for reception of data, such as UART data, via DMA. DMA may be used to continuously transfer data from reception buffer 114 to memory 118 of microcontroller 114 implementing the communication protocol. DMA is normally running to receive incoming frame data. The hardware break character detection signal at the output of break character detection circuit 112 interrupts microcontroller 116 which in turn stops the DMA transfer and runs machine readable instructions that process the received frame. In examples, the frame length of the received message may be derived from the DMA byte count when stopping the DMA transfer. With this solution, microcontroller 114 is relieved from interacting with the hardware

interface, such as the UART peripheral, until all the content of the incoming frame is available in the microcontroller memory 118. Additionally, the microcontroller 116 may process data from its memory very quickly, thereby allowing fast response times.

[00063] In examples, all communications take place in the form of a request/response transaction, where the master node sends the request and the slave node provides the response. The protocol may keep things simple by having a fixed frame size for the requests and the response. In examples, two combinations of frame sizes are allowed, break character not included:

READ TRANSFER: the request is 3-byte long and the response is 7-byte long

WRITE TRANSFER: the request is 7-byte long and the response is 3-byte long.

[00064] Thus, examples allow for a robust reception process on both the slave node side and the master node side since both accept as a valid frame one that matches a valid frame length and do not further process others. This permits many cases of bad frame reception due to electrical or hardware issues to be filtered out.

[00065] In examples, the frame contains a checksum field to validate the integrity of the data. The checksum field may be a CRC, cyclic redundancy check, field. In detail, a one-byte CRC-8 value may be included as the last byte in the frame, right before the break character. CRC-8 allows to detect up to two non-consecutive data bit errors or up to seven consecutive data bit errors in a frame. Additionally, CRC-8 may be efficiently implemented using machine readable instructions doing a lookup operation and a bitwise XOR operation per each byte in the frame. In addition, use of an 8-bit length checksum may limit the per-frame checksum overhead. Thus, using such a checksum is efficient for

a communication protocol which uses short but frequent messages and which is focused on low-end microcontrollers with reduced computational capabilities.

[00066] Examples of the present disclosure further permit communication error recovery. In examples, the master node may detect that a communication error occurred in a transaction by the lack of a successfully received response. The master node may thus perform a retransmission of the unacknowledged request. In doing so, care is to be taken at the slave node side to detect and discard possible duplicate transmission requests, which may occur when the request is successfully received at the slave node side but the response is not successfully received at the master node side. Thus, in examples, the master node includes in the header of the request a retransmission counter that increments with each retransmission. The slave node may therefore store a copy of the last successfully received request frame, and discard any new request that is identical to the previous and has a larger retransmission count.

[00067] In other examples, the master node may proactively send N repetitions of the same request message to the slave nodes. The slave node may discard duplicate requests using the previously mentioned mechanism. This repetition scheme may be effective in having a predictable bandwidth requirement and latency, which may make it more suitable when it comes to network dimensioning. Additionally, as opposed to the use of retransmissions, the master node implementation is stateless, and hence may be efficiently implemented on hardware such as FPGAs.

[00068] In examples, the MSCP prescribes a specified response time to be respected by each slave node. The response time is defined as the time duration between the end of the break character corresponding to a request frame as seen by a slave node on the request line and the beginning of the response frame transmission by the same slave node on the response line.

[00069] In examples, compliance with the specified response time may be realized in the slave node as follows. Whenever a request frame is correctly

received by its intended slave node, the slave node in question starts a hardware timer set to the value of the response time. Then the slave node proceeds to perform the operations related to the received request. When completed, it generates the response frame and sets a flag identifying the response frame as ready for transmission. When the response timer elapses, a routine is invoked which checks the response frame ready flag. If the flag is set, the response is transmitted and the transaction is considered as ended successfully from the slave node side. On the other hand, if due to an error in machine readable instructions or due to unexpected computational latency the response flag is not set by the time the timer elapses, the response is not set, and a timeout error is registered on the slave node side.

[00070] In examples, the communication protocol may provide control of the watchdog on the slave nodes. While the master node appeases the watchdog, the slave node is assured of connectivity with the master node. Control of the watchdog service on the slave node may comprise two registers, a watchdog enable register and a watchdog timeout register. The watchdog timeout register is used to query and set the timeout value of the watchdog on the slave node. The watchdog enable register is used to enable, disable, and appease the watchdog.

[00071] Examples of the present disclosure provide a method of communicating in a communication system as described herein. Fig. 8 shows a flowchart of such a method. At 200, an incoming frame is received and an end of the incoming frame is detected by the hardware interface using a break character. At 202, the node checks whether the incoming frame has a fixed frame length. If the incoming frame has the fixed frame length, the frame is further processed and if the incoming frame does not have the fixed frame length, the incoming frame is not further processed, 204.

[00072] Examples of the present disclosure provide a communication system as described including a master node and slave nodes. Examples of the present disclosure provide a master node to be used in such a communication system

and examples of the present disclosure provide a slave node to be used in such a communication system. Examples of functionalities of a slave node are now described referring to the flow chart of Fig. 9, and examples of functionalities of a master node are now described referring to the flow chart in Fig. 10.

[00073] In examples, each slave node is to perform a reception process as shown in Fig. 9. At 300, the break character is received on the request line. The break character indicates the end of a frame and, therefore, at 302, receiving DMA is stopped. At 304, it is checked whether the received frame has one of valid fixed frame lengths, which, in examples, may be a frame length of 3 bytes or 7 bytes. If the received frame does not have the fixed frame length of 3 byte or 7 bytes, but any other frame length, a form error is recognized, 306. If the received frame has a frame length of 3 byte, the frame is a read request, 308, and if the received frame has a frame length of 7 byte, the frame is a write request, 310. In case of a read request, the CRC value of the frame is fetched from the third byte, 312, and in case of a write request, the CRC value of the frame is fetched from the seventh byte, 314. At 316, a new CRC value is calculated from the frame, such as from all bytes except for the CRC value of the frame. At 316 it is checked whether the new CRC value matches the CRC value of the frame. If the CRC values match the checksum is validated successfully and if the CRC values do not match the checksum is not validated successfully. If the CRC values match, the frame is sent to a link layer for processing, 320. If the CRC values do not match, an CRC error is recognized, 322. Upon recognizing a form error at 306 the receiving DMA is started again and the reception process ends at 324. Upon recognizing a CRC error at 322 the receiving DMA is started again and the reception process ends at 324. Upon sending the frame to the link layer, the receiving DMA is started again at 322 and the reception process ends at 324.

[00074] In examples, each master node is to perform a process as shown in Fig. 10. At 400, the master node sends a read request or a write request on the request line. Corresponding transmissions may be requested by upper layers of

the master node. At 402, a retransmission counter R is set to 0. At 404, the read request or the write request is sent on the request line, such as the MOSI line shown in Fig. 2. At 406 a timeout is started. At 408 and 410 it is determined whether a break character is received on the response line, such as the MISO line in Fig. 2, before the timeout expired. If so, a response to the request was received in time and the process jumps to 412. At 412 the receiving DMA stops. If a response was not received in time, i.e. the timeout expired before receiving a response, the retransmission counter R is increased by one at 414. The retransmission counter is compared to a threshold Rlimit and it is checked at 416 whether the retransmission counter exceeds the threshold Rlimit. If the retransmission counter does not exceed the threshold, retransmission of the frame takes place at 404. In the retransmission, the increased retransmission counter is included in the frame.

[00075] After stopping the receiving DMA at 412, it is determined whether the received frame has the correct frame length at 418. To be more specific, it is determined whether the DMA byte count matches the response size for the transmitted request. If not, a form error is recognized at 420. If it is determined that the received frame has the correct frame length at 418, the CRC value is fetched from the frame at 422. In examples, the CRC value is fetched from the last byte of the frame, i.e. the byte preceding the break character. At 424 a new CRC value is calculated using the received frame, such as using all bytes of the frame except for the CRC value of the frame. At 426 it is checked whether the CRC value of the frame and the new CRC value match. If the CRC values match the checksum is validated successfully and if the CRC values do not match the checksum is not validated successfully. If the CRC values do not match, a CRC error is recognized at 428. If the CRC values match, at 430 the response frame is further processed by sending the response frame content to upper layers of the master node for processing. Thereupon, the receiving DMA is started at 432 and the process ends at 434.

[00076] Recognizing a form error at 420 or a CRC error at 428 means that the response was not received successfully. Thus, upon recognizing either a form error at 420 or a CRC error at 428, the receiving DMA is restarted at 436. Upon restarting the receiving DMA at 436, the process jumps to 414 and the retransmission counter is increased by 1. The increased retransmission counter is again compared to the threshold Rlimit.

[00077] If the comparison at 416 reveals that the retransmission counter exceeds the threshold, a communication error is reported to upper layers of the master node at 418 and the process ends at 434.

[00078] Accordingly, examples of the present disclosure permit communication between a master node and slave nodes in an easy manner while maintaining the possibility of recognizing errors and recovering from errors. In particular, examples permit reduction of workload upon recognizing that a received frame does not have the correct frame length.

[00079] Examples relate to a non-transitory machine-readable storage medium encoded with instructions executable by a processing resource of a computing device to perform methods described herein.

[00080] Examples described herein may be realized in the form of hardware, machine-readable instructions or a combination of hardware and machine-readable instructions. Any such machine-readable instructions may be stored in the form of volatile or non-volatile storage such as, for example, a storage device, such as a ROM, whether erasable or rewritable or not, or in the form of memory, such as, for example, RAM, memory chips, device or integrated circuits or an optically or magnetically readable medium, such as, for example, a CD, DVD, magnetic disk or magnetic tape. The storage devices and storage media are examples of machine-readable storage, that are suitable for storing a program or programs that, when executed, implement examples described herein.

[00081] All of the features disclosed in the specification including any accompanying claims, abstract and drawings, and/or all the features of any method or progress described may be combined in any combination including any claim combination, except combinations where at least some of such features are mutually exclusive. In addition, features disclosed in connection with a system may, at the same time, present features of a corresponding method, and vice versa.

[00082] Each feature disclosed in the specification including any accompanying claims, abstract and drawings may be replaced by other features serving the same, equivalent or a similar purpose, unless expressly stated otherwise. Thus, unless expressly stated otherwise, each feature disclosed is one example of a generic series of equivalent or similar features.

[00083] The foregoing has described the principles, examples and modes of operation. However, the teachings herein are not be construed as being limited to the particular examples described. The above-described examples are to be regarded as illustrative rather than restrictive, and it is to be appreciated that variations may be made in those examples by workers skilled in the art without departing from the scope of the following claims.

Claims

What is claimed is:

1. A communication system comprising:

a master node and slave nodes,

wherein the master node and the slave nodes are coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node,

wherein each node comprises a hardware interface to receive an incoming frame, the hardware interface to detect an end of the incoming frame using a break character,

wherein each node is to check whether the incoming frame has a correct frame length, to further process the incoming frame if the incoming frame has the correct frame length, and not to further process the incoming frame if the incoming frame does not have the correct frame length.

2. The communication system of claim 1, wherein the hardware interface is to detect the end of the incoming frame using the break character formed by a number of at least ten bits having the same level different from an idle level of the request line and the response line.

3. The communication system of claim 1, wherein further processing the incoming frame includes validating a checksum of the incoming frame by the node and any processing following the validation.

4. The communication system of claim 1, wherein further processing the incoming frame by a slave node comprises sending a response to the master node if the incoming frame is addressed to that slave node, and forwarding the

incoming frame to the next stage of the daisy chain if the incoming frame is not addressed to that slave node.

5. The communication system of claim 1, wherein each request frame is either of a write request type or a read request type, wherein the frame length of a request frame of the write request type is larger than the frame length of the read request type, and wherein each response frame is either of a write response type or a read response type, wherein the frame length of a response frame of the read response type is larger than the frame length of a response frame of the write response type.

6. The communication system of claim 1, wherein the hardware interface is to transfer the incoming frame to a memory using a direct memory access.

7. The communication system of claim 1, wherein the master node is to retransmit a request frame once or N times and wherein the master node is to increase a count in a header of this request frame every time the request frame is retransmitted.

8. The communication system of claim 7, wherein the master node is to retransmit the request frame upon expiration of a timeout since transmitting the request frame without receiving a response on the response line, and/or if the incoming frame does not have the correct frame length, and/or if a checksum in the incoming frame is not validated successfully.

9. The communication system of claim 7, wherein a slave node which received a request frame successfully is to discard any retransmission of the request frame having a higher count.

10. The communication system of claim 1, wherein each slave node comprises a hardware timer and a slave node addressed in a request frame is

to transmit a response to the request frame within a predetermined time window using the hardware timer.

11. A slave node for a communication system comprising a master node and slave nodes coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node,

wherein the slave node comprises a hardware interface to receive an incoming request frame, the hardware interface to detect an end of the incoming frame using a break character,

wherein the slave node is to check whether the incoming request frame has a correct frame length, to further process the incoming frame if the incoming frame has the correct frame length, and not to further process the incoming frame if the request frame does not have the correct frame length.

12. The slave node of claim 11, wherein further processing the frame comprises sending a response to the master node if the frame is addressed to that slave node, and forwarding the frame to the next stage of the daisy chain if the frame is not addressed to that slave node.

13. A master node for a communication system comprising the master node and a plurality of slave nodes coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node,

wherein the master node comprises a hardware interface to receive an incoming response frame, the hardware interface to detect an end of the incoming frame using a break character,

wherein the master node is to check whether an incoming response frame has a correct frame length, to further process the incoming response frame if the incoming response frame has the correct frame length, and not to

further process the incoming response frame if the response frame does not have the correct frame length.

14. The master node of claim 13, wherein the master node is to retransmit the request frame once or N times and to increase a count in a header of the request frame every time the request frame is retransmitted.

15. A method of communicating in a communication system comprising a master node and slave nodes, wherein the master node and the slave nodes are coupled to each other in a daisy chain configuration, the daisy chain configuration including a request line to transmit request frames from the master node to the slave nodes and a response line to transmit response frames from the slave nodes to the master node, wherein each node comprises a hardware interface at which the frames are received, and wherein each frame has a frame type and a correct frame length given by the frame type, the method comprising at one of the nodes:

receiving an incoming frame and detecting, by the hardware interface, an end of the incoming frame using a break character;

checking whether the incoming frame has the correct frame length,

further processing the incoming frame if the frame has the correct frame length, and

not further processing the incoming frame if the frame does not have the correct frame length.

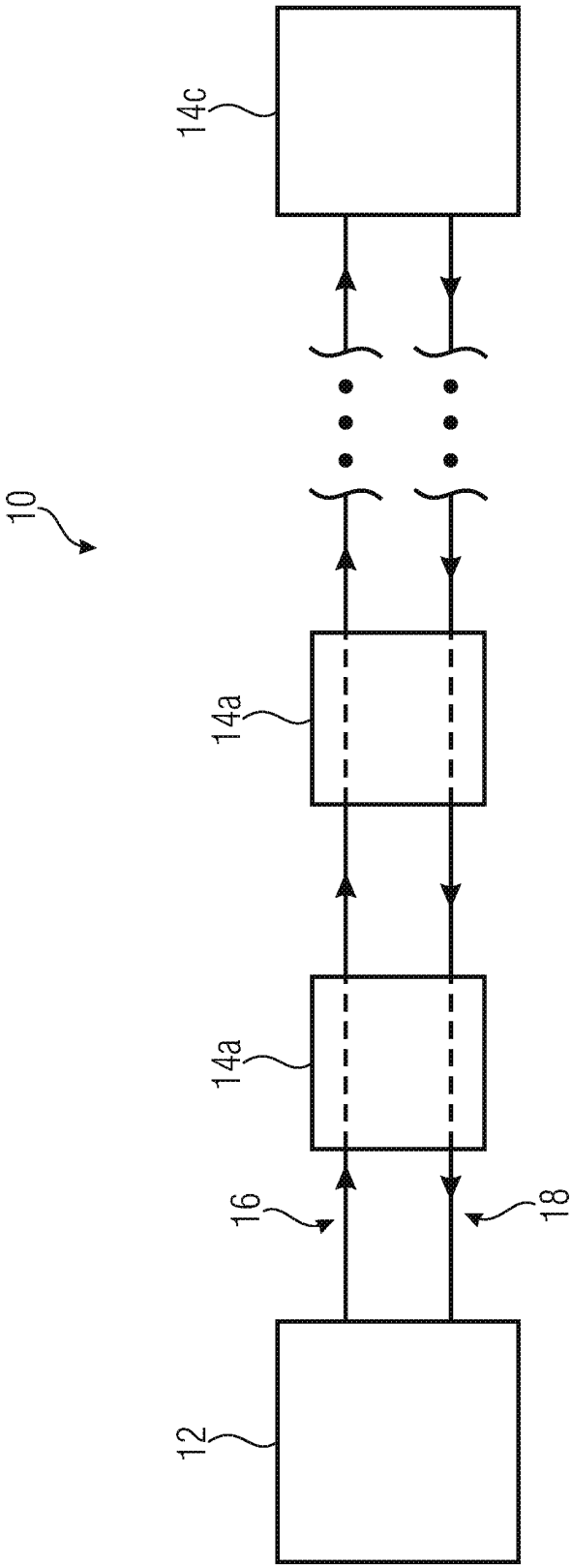

Fig. 1

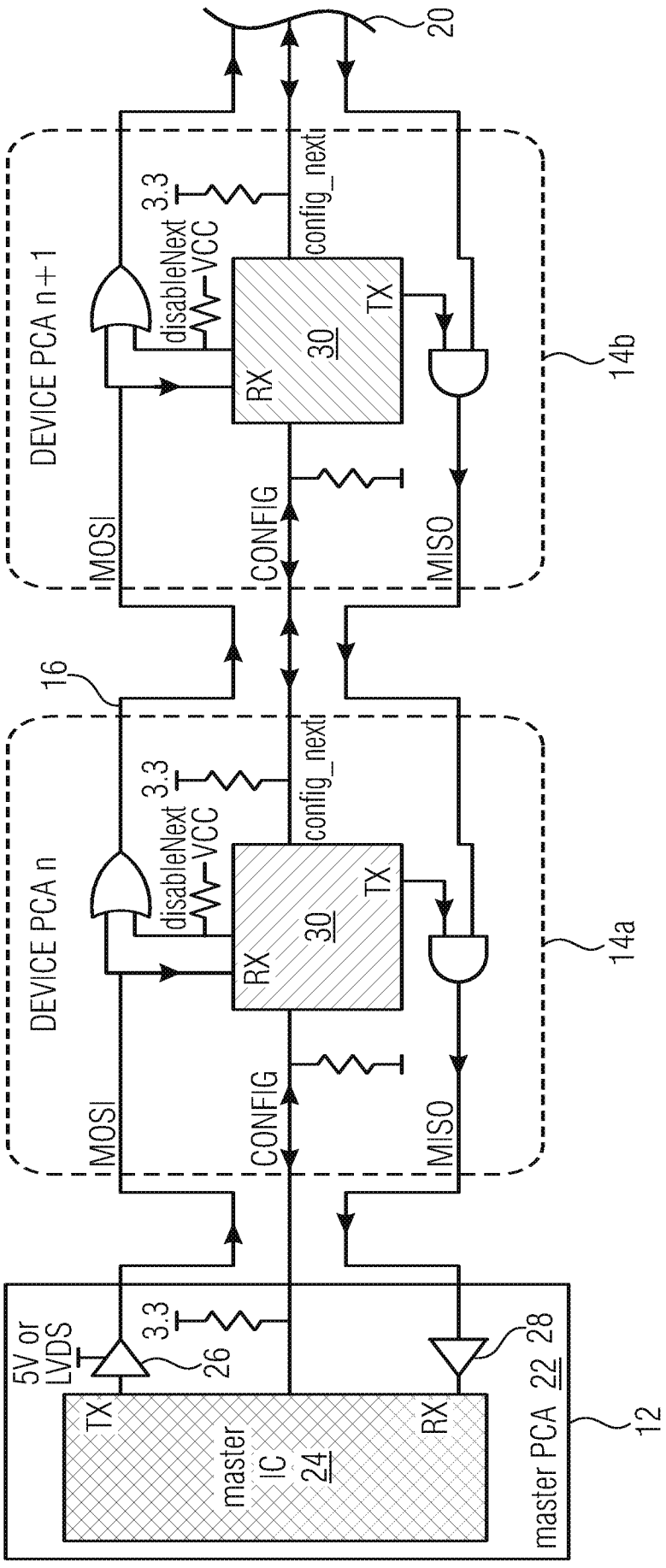
20
DEVICE PCA n+1
3.3
disableNext
VCC
config_next
RX
30
TX
14b
MOSI
CONFIG
MISO
16
DEVICE PCA n
3.3
disableNext
VCC
config_next
RX
30
TX
14a
MOSI
CONFIG
MISO
5V or LVDS
3.3
26
TX
master IC 24
RX
28
master PCA 22
12

Fig. 2

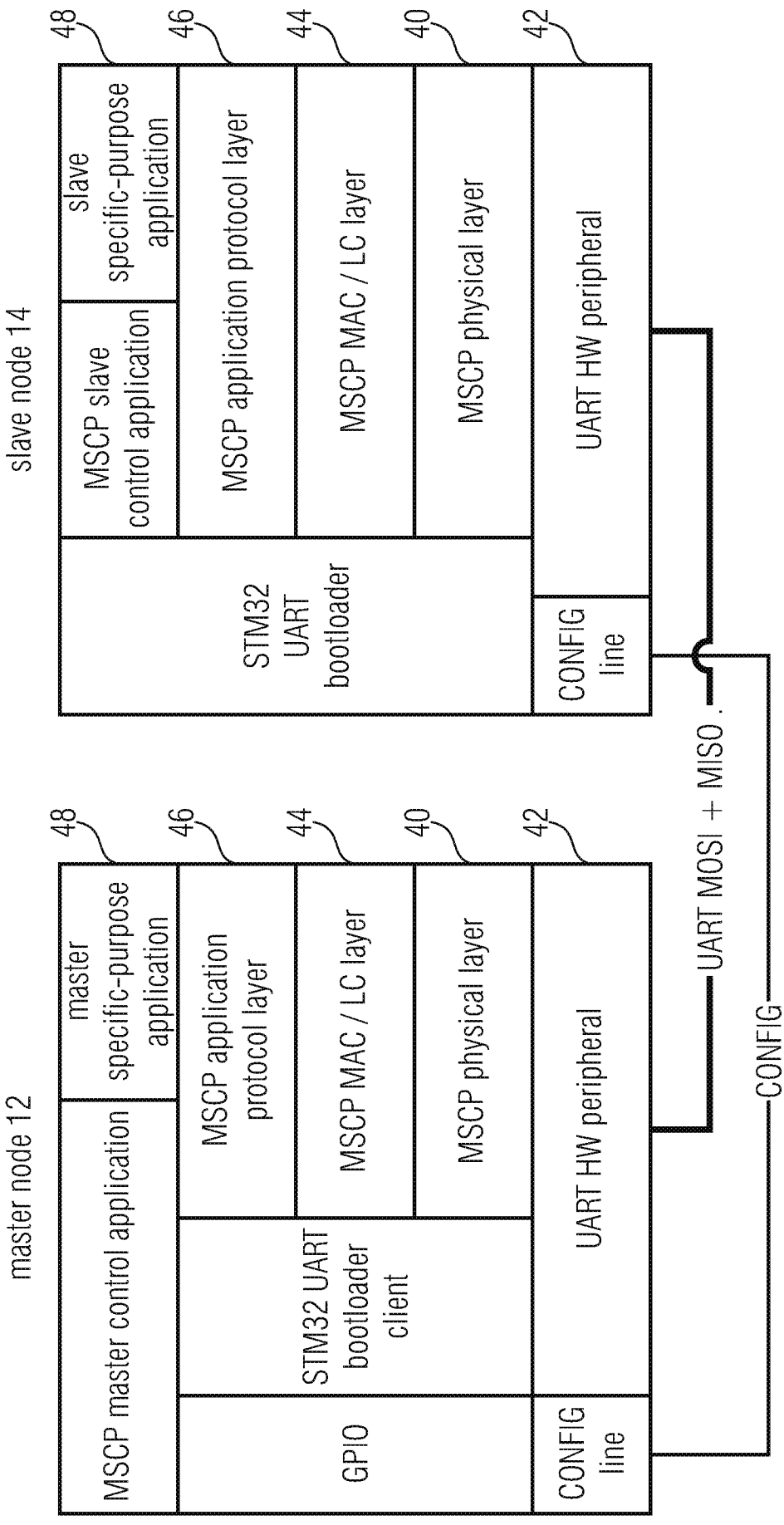
slave node 14
48
46
44
40
42
slave specific-purpose application
MSCP slave control application
MSCP application protocol layer
MSCP MAC / LC layer
MSCP physical layer
UART HW peripheral
STM32 UART bootloader
CONFIG line
UART MOSI + MISO
CONFIG
master node 12
48
46
44
40
42
master specific-purpose application
MSCP master control application
MSCP application protocol layer
MSCP MAC / LC layer
MSCP physical layer
UART HW peripheral
STM32 UART bootloader client
GPIO
CONFIG line

Fig. 3

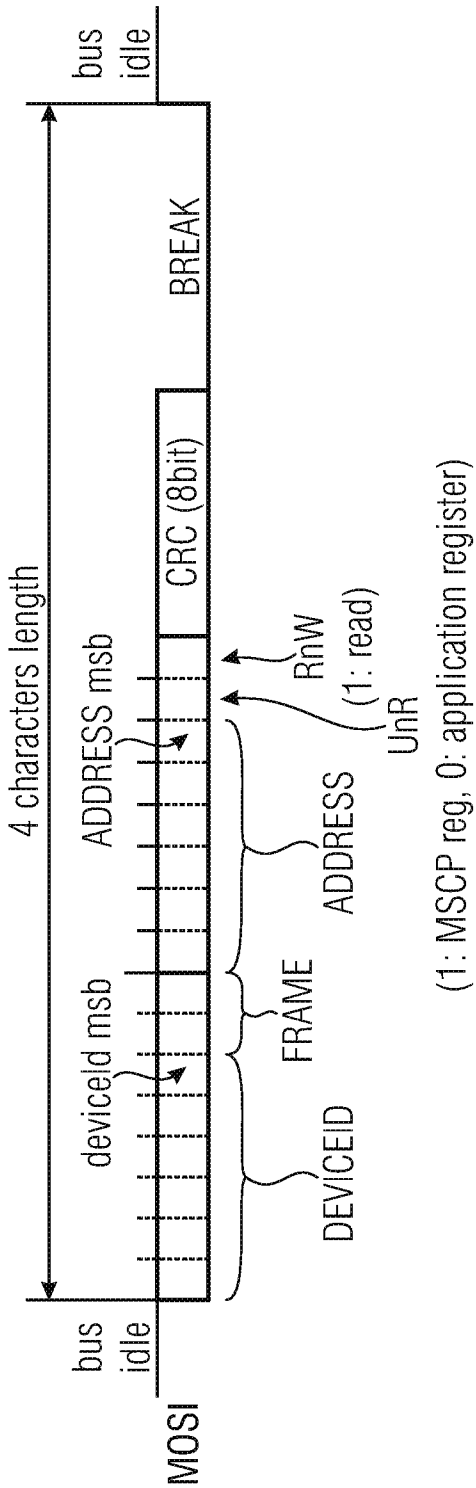
MOSI
bus idle
4 characters length
deviceId msb
ADDRESS msb
CRC (8bit)
BREAK
bus idle
DEVICEID
FRAME
ADDRESS
UnR
RnW
(1: read)
(1: MSCP reg, 0: application register)

Fig. 4A

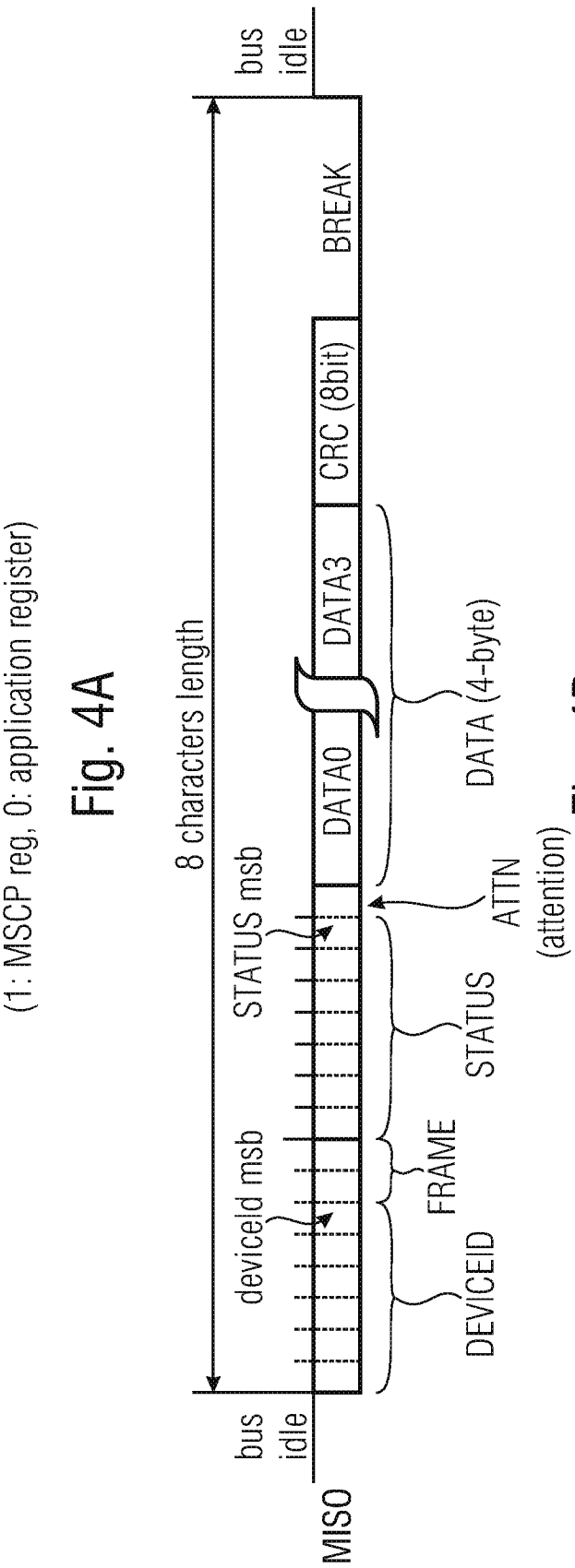
MISO
bus idle
8 characters length
deviceId msb
STATUS msb
DATA0
DATA3
CRC (8bit)
BREAK
bus idle
DEVICEID
FRAME
STATUS
ATTN
(attention)
DATA (4-byte)

Fig. 4B

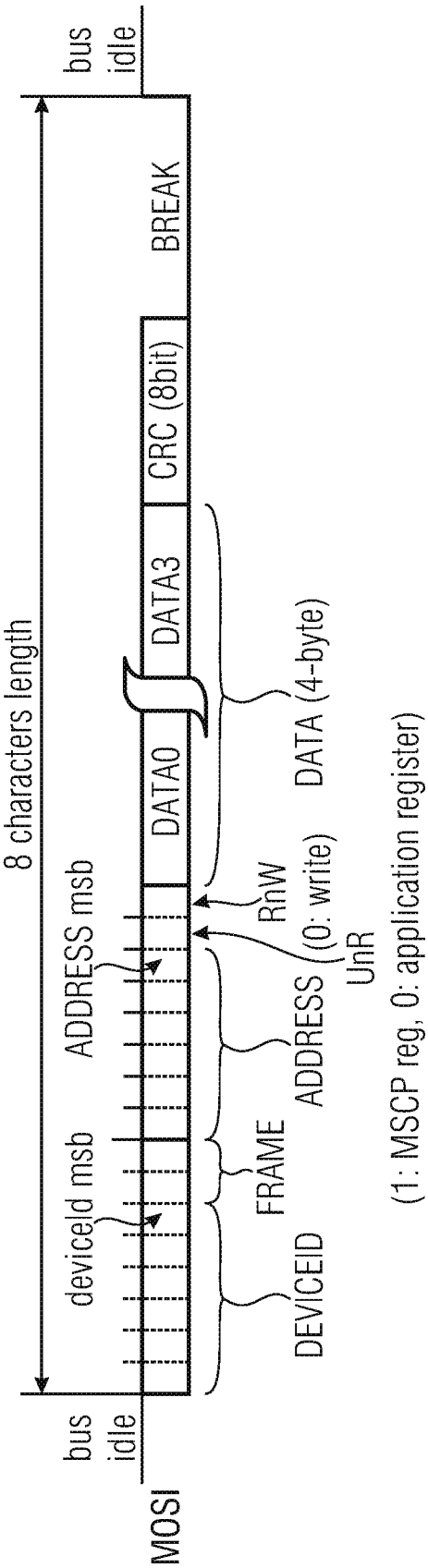
MOSI
bus idle
deviceId msb
DEVICEID
FRAME
ADDRESS msb
ADDRESS
RnW
(0: write)
UnR
(1: MSCP reg, 0: application register)
DATA0
DATA3
DATA (4-byte)
CRC (8bit)
BREAK
bus idle
8 characters length

Fig. 5A

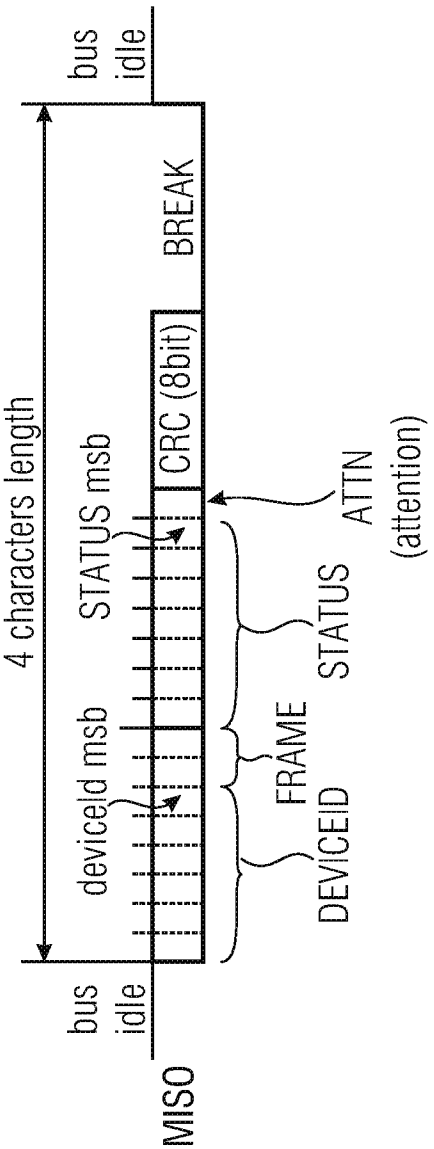
MISO
bus idle
deviceId msb
DEVICEID
FRAME
STATUS msb
STATUS
ATTN
(attention)
CRC (8bit)
BREAK
bus idle
4 characters length

Fig. 5B

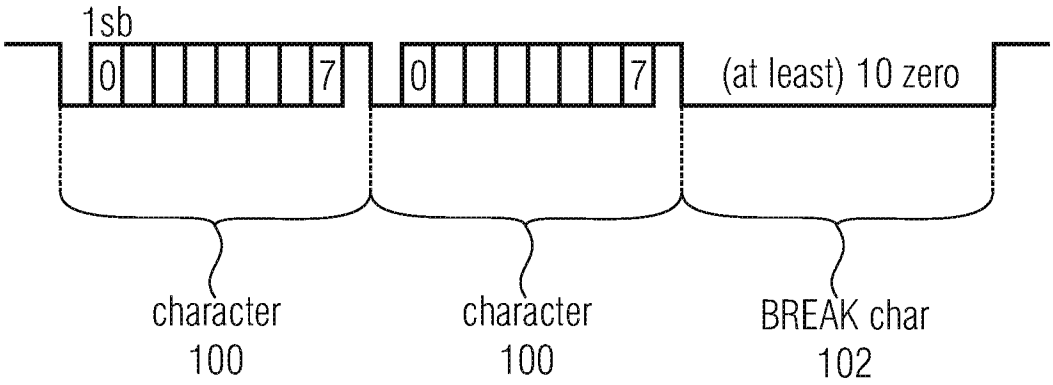
1sb
0
7
0
7
(at least) 10 zero
character
100
character
100
BREAK char
102

Fig. 6

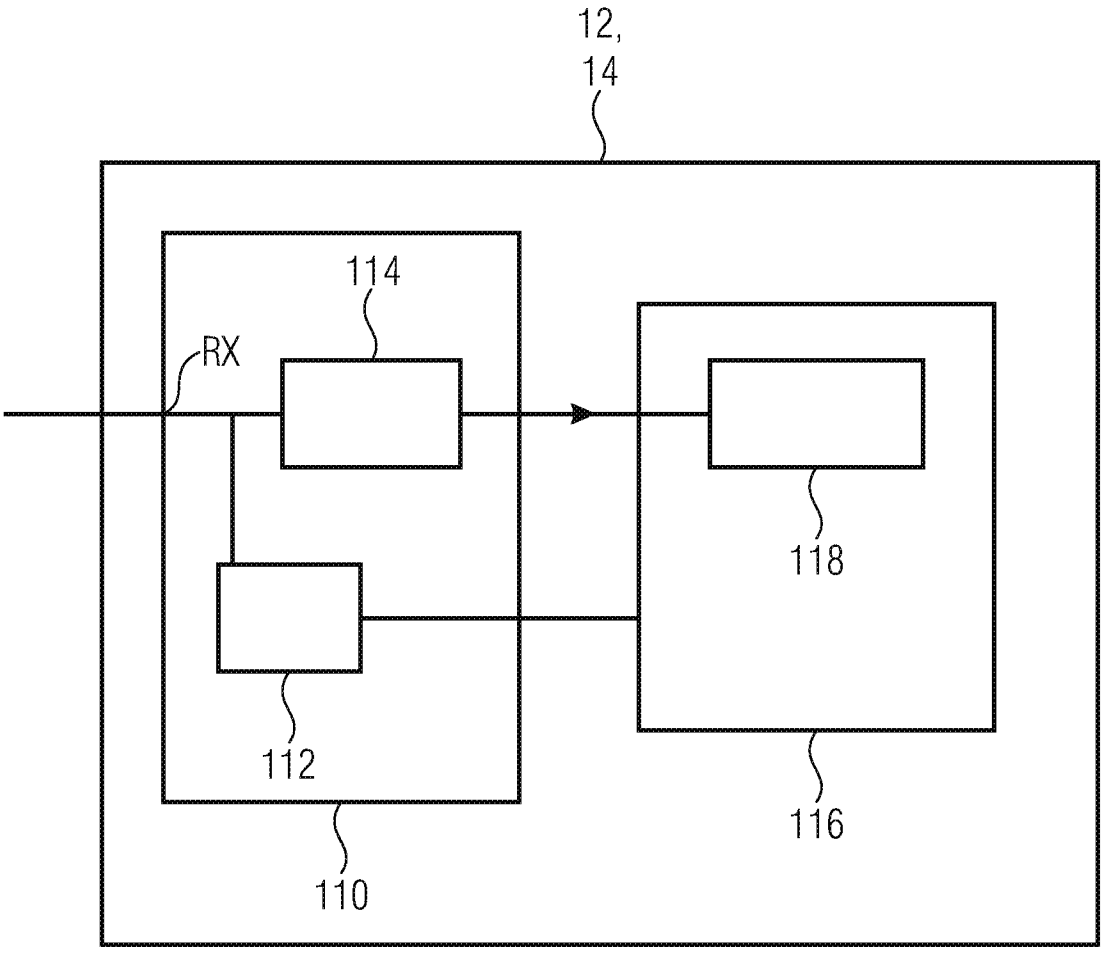
12,
14
114
RX
118
112
116
110

Fig. 7

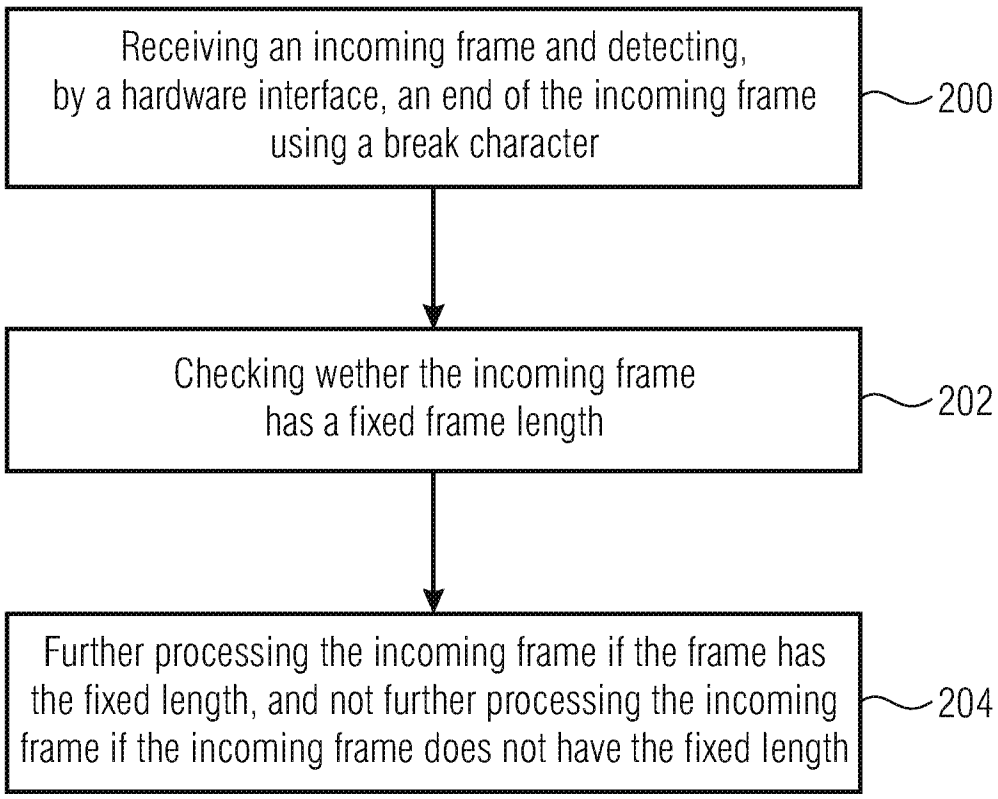
Receiving an incoming frame and detecting, by a hardware interface, an end of the incoming frame using a break character
200
Checking wether the incoming frame has a fixed frame length
202
Further processing the incoming frame if the frame has the fixed length, and not further processing the incoming frame if the incoming frame does not have the fixed length
204

Fig. 8

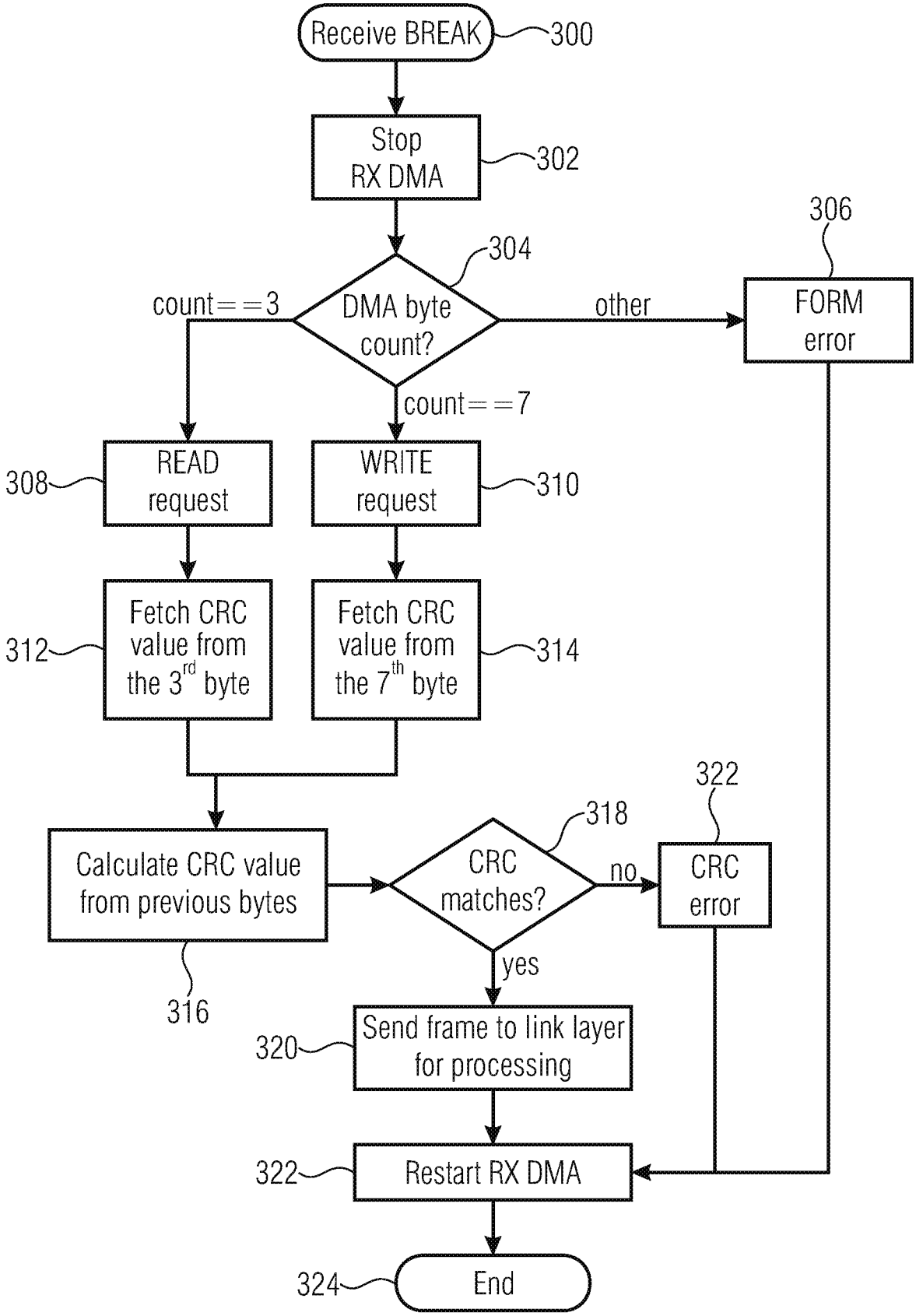
Receive BREAK
300
Stop RX DMA
302
304
DMA byte count?
count==3
count==7
other
306
FORM error
READ request
308
WRITE request
310
Fetch CRC value from the 3rd byte
312
Fetch CRC value from the 7th byte
314
Calculate CRC value from previous bytes
316
318
CRC matches?
no
yes
322
CRC error
Send frame to link layer for processing
320
Restart RX DMA
322
End
324

Fig. 9

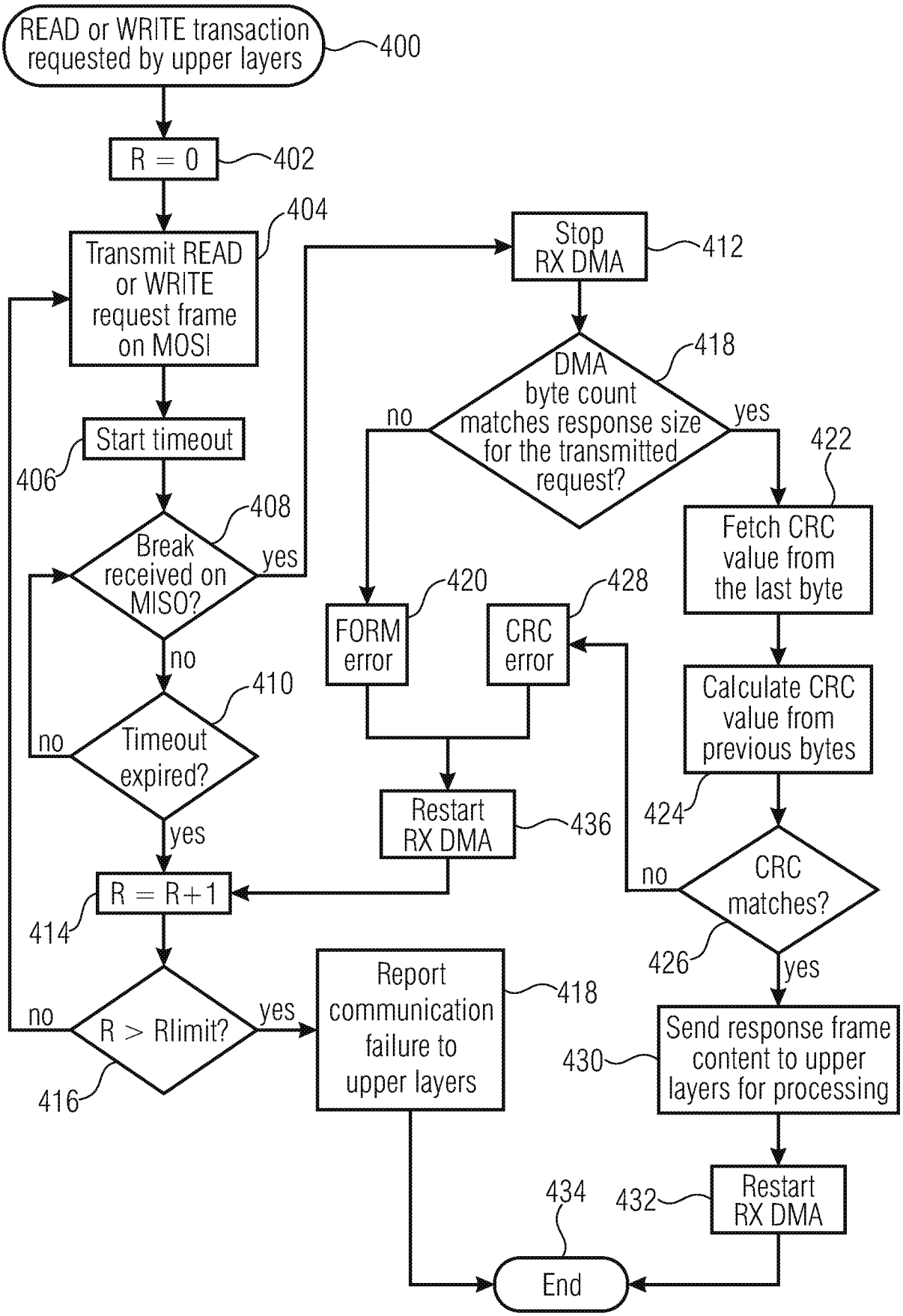
READ or WRITE transaction requested by upper layers
400
R = 0
402
Transmit READ or WRITE request frame on MOSI
404
Start timeout
406
Break received on MISO?
408
yes
no
Timeout expired?
410
no
yes
R = R+1
414
R > RIimit?
416
no
yes
Report communication failure to upper layers
418
End
Stop RX DMA
412
DMA byte count matches response size for the transmitted request?
418
no
yes
FORM error
420
CRC error
428
Restart RX DMA
436
Fetch CRC value from the last byte
422
Calculate CRC value from previous bytes
424
CRC matches?
426
no
yes
Send response frame content to upper layers for processing
430
Restart RX DMA
432
434

Fig. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 2018/053287

A. CLASSIFICATION OF SUBJECT MATTER

H04L 12/24 *(2006.01)*
H04L 12/403 *(2006.01)*

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PatSearch (RUPTO internal), Espacenet, DWPI, PAJ, USPTO

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP 2009-239429 A (MITSUBISHI) 15.10.2009, abstract, fig.1, [0004], [0008]	1, 4, 6, 7-15
A		2,3,5
Y	EP 1312185 B1 (KOSTOFF et al.) 05.03.2014, [0002], [0004], [0008], [0009], [0013], [0014], [0029], [0038], [0088], [0090], [0091], [0107], [0108]	1, 4, 6, 7-15
A	JP 2017-005364 A (TOSHIBA) 05.01.2017, abstract	1-15

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
05 March 2019 (05.03.2019)	20 June 2019 (20.06.2019)

Name and mailing address of the ISA/RU:
Federal Institute of Industrial Property,
Berezhkovskaya nab., 30-1, Moscow, G-59,
GSP-3, Russia, 125993
Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37

Authorized officer
V. Aleksandrov

Telephone No. (499) 240-25-91