Two approaches to automatic configuration of RS-485 network

Nikola Cvetković, Pavle Milenković, Nenad Jovičić and Vladimir Rajović

***Abstract*—The purpose of this paper is to provide an overview of common approaches to automatic configuration of half-duplex RS-485 network, as well as to introduce two alternative methods of automatic slave address configuration in a network. Described mechanisms will be analyzed and compared in terms of hardware and software complexity, while taking into consideration system robustness and implementation feasibility.**

***Index Terms*—RS-485, automatic configuration, differential bus, communication network**

I. Introduction

The interface commonly known as RS-485 represents an electrical standard used in serial communication systems. It only defines electrical characteristics of drivers and receivers connected into a network [1], thus leaving the opportunity of using various standardized or user-defined data communication protocols. Some of the examples of frequently used protocols include Modbus, used in industrial settings, and BACnet, often applied in automation of buildings and other monitoring applications [2]. RS-485 is a balanced differential electrical bus, either full-duplex or half-duplex, supporting up to 32 unit loads, where each unit load represents an impedance of approximately 12 kΩ. Fig. 1 shows an example of a common balanced system. It consists of a driver D and a receiver R. A termination resistor R_T is used on the input of the receiver, to match the input impedance of the lines.

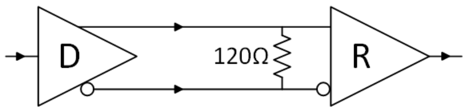


Fig. 1. A common balanced system

Since the lines of the bus are balanced, meaning they have equal impedances along their length and equal impedances with respect to ground, noise induced in the conductors and their electromagnetic radiation is minimized. Drivers' minimal differential output voltage has to be greater than 1.5 V across a load resistance of 54 Ω, while the minimal detectable differential input voltage of receivers is 200 mV. Described characteristics allow for a conventional transmission speed of up to 10 Mb/s and the maximum range of 1200 meters. When designing a system, an empirical relation for transmission speed and bus length ratio has to be taken into consideration [3]. Although the standard recommends 10 Mb/s, today's fast interface circuits are optimized for data rates of up to 50 Mb/s [4].

All the approaches for the automatic network configuration that will be analysed in this paper use a master/slave model of communication, in which only the master can initiate communication with the slave. The master can either send a broadcast address in order to address all the nodes in the network, or it can address individual slave by sending a specific address.

RS-485 standard supports various network topologies, the simplest of which is point-to-point connection (Fig. 2).

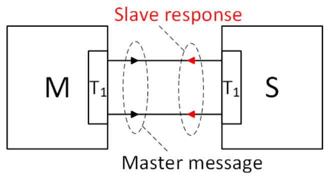


Fig. 2. Point-to-point topology

Adding additional network nodes to the topology mentioned above requires insertion of junction boxes. This modification, referred to as Backbone with Stubs, consists of a main differential bus called a backbone, branching off to the slaves with stubs (Fig. 3). The described topology is one of the two most commonly used methods for connection of the nodes.

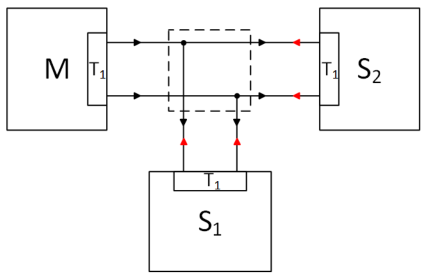


Fig. 3. Backbone with stubs topology

The second most frequently used connection method is daisy-chain topology (Fig. 4), which requires each two consec-

Nikola Cvetković is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: cn203094m@student.etf.bg.ac.rs).

Pavle Mileknović is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: mp203016mm@student.etf.bg.ac.rs).

Nenad Jovičić is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: nenad@etf.bg.ac.rs).

Vladimir Rajović is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: rajo@etf.bg.ac.rs).

utive nodes to be directly connected to each other via dedicated connection ports.

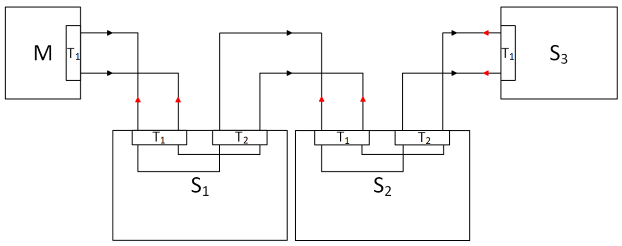


Fig. 4. Daisy-chain topology

When designing a network, special care has to be taken in order to provide a unique address for each node. In conventional address assigning methods, problems occur in manual work cost, hardware and software complexity, and inconvenience in network modification, including faulty node removal and network expansion. Without loss of generality, the main focus of the analysis will be on half-duplex RS-485 bus. Additionally, communication is performed in a traditional master-slave manner and information about relative physical position of each node has to be available at all times.

II. Common Slave Addressing Approaches

In practice, there are three widely used methods for node address configuration, which will be covered in more detail in the following subsections. These include different hardware address assignments, firmware hard coding of an address and software address assignment upon addition of a new node, one by one. The common characteristic of all these methods is that they are all in need of node configuration prior to address assignment.

A. Hardware Address Assignment

The simplest way to assign an address to a slave from a software perspective is to provide the hardware for address configuration. This can be achieved by using dual in-line package (DIP) switches, rotary switches, on-board jumpers, or by directly connecting every slave's address pin to either high or low logic levels. A typical configuration using a DIP switch is represented by Fig. 5.

In spite of being easy to implement, relatively cheap to design and in some applications easily reconfigurable, this method has considerable setbacks. This system is prone to human error, since all switches/jumpers rely on manual configuration, increasing the possibility of multiple devices having the same address. Additionally, in an environment where network nodes are not easily accessible, modification of the addresses might pose an issue. In some cases, major flaw of described approach can be the price of manual work required for network initialization and modification. Despite having the mentioned limitations, this approach is frequently found in commercial applications [5], [6].

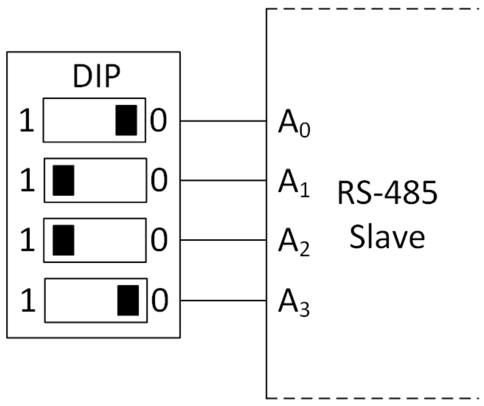


Fig. 5. Address configuration using a DIP switch

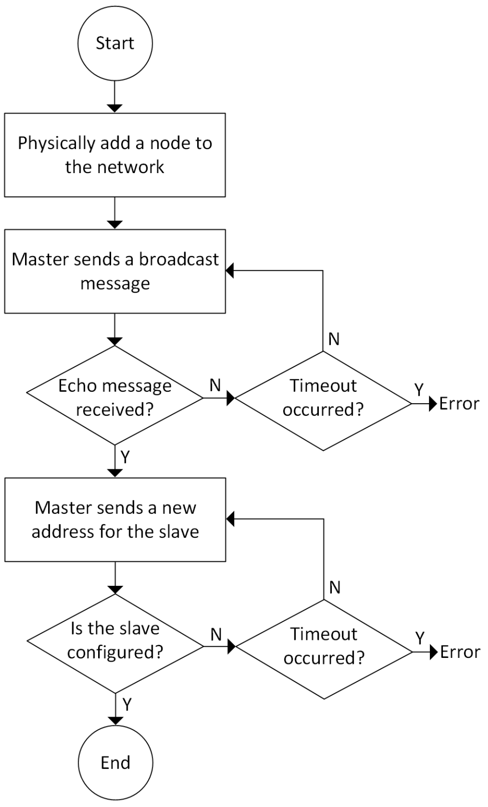
B. Firmware Hard Coding of an Address

Another method is the hard coding of an address inside the slave firmware. This approach results in a simpler and cheaper design, as it does not require any additional hardware. Since it does not require a direct human contact during deployment in order to configure the slave, the system is less prone to human error than the previous approach. However, the main weakness of this method lies in the effort to write and maintain different versions of firmware for each node in the network. In terms of network extension and modification, nothing is improved compared to the hardware assignment method.

C. One by One Network Expansion

If hardware modification of the nodes is not possible, different firmware versioning can be avoided by gradual expansion of the network, one node at the time. A common algorithm of RS-485 bus initialization is represented by Fig. 6.

At the beginning of the initialization of the network, every slave is configured to have the same default address, e.g. address 0x00, while the initial topology resembles point-to-point communication (Fig. 2). Master sends a broadcast message, addressing all slaves with the default address. Since the nodes are being added to the network one by one, at any given moment, there will be no more than one device with the default address. The node with the default address sends an echo message, requesting the new address from the master. Finally, the master sends a message containing the new address of the slave and waits for the confirmation of the message, checking if the slave is configured properly. Master can be configured to send a broadcast message periodically, or by user command upon connection of a new node. Some implementation of a waiting/timeout loop should be considered, since various errors in a network can result in improper signal reception, either on master or slave side of the connection. Another advantage of this method is the simplicity of adding new nodes into a network - no additional work to adjust the software or configure the switches is required, apart from physically connecting the nodes one after another. Although relatively simple in terms of hardware and software complexity from the



Start
Physically add a node to the network
Master sends a broadcast message
Echo message received?
N
Timeout occurred?
N
Y
Error
Y
Master sends a new address for the slave
Is the slave configured?
N
Timeout occurred?
N
Y
Error
Y
End

Fig. 6. One by one initialization algorithm

slave side, this approach requires a long initialization process, which results in a costly manual work.

III. Proposed Solutions

In the previous sections, the internal structure of network nodes was implemented using a single RS-485 transceiver. A typical example of an RS-485 transceiver can be observed in the Fig. 7.

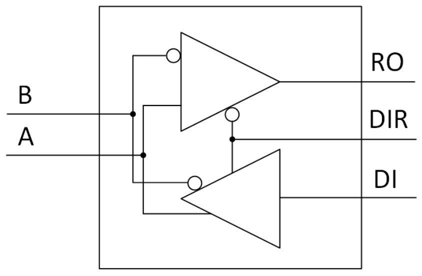


Fig. 7. Typical RS-485 transceiver

By designing the network nodes using additional RS-485 transceiver, new methods for automatic address assignment can be formulated.

A. Utilizing Network Topology for Slave Addressing (Domino Method)

Proposed structure of a network node is given in the Fig. 8. This solution offers a flow of information through the slave, by

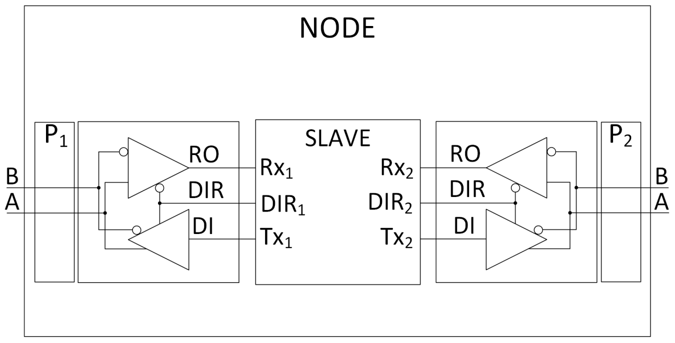


Fig. 8. Node structure for the Domino method

making use of two serial ports. The line topology obtained by connecting multiple slaves with previously presented structure is shown in Fig. 9. Nodes are denoted by N_i , $i = \overline{0, n-1}$, while each node has two ports, P_1 and P_2.

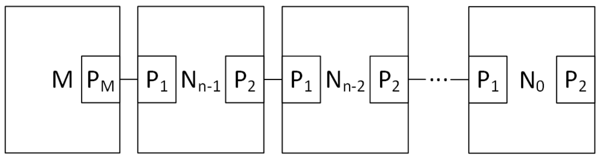


Fig. 9. A proposed network topology for the Domino method

The basic principle of the proposed solution is that there is a predetermined address that triggers the responses from every slave, similarly to the broadcast address in any multiple-access communication network. In a given example, the value of predetermined address is 0x00. The system initialization phase starts with determining the number of nodes in the network. Depending on the expected number of nodes, either incremental or decremental approach can be used. Perhaps more intuitive is the incremental approach, used for small-scale networks. The algorithm starts by master sending a test message over P_M port, containing the predetermined address, which stores the information about nodes counted thus far, in this case 0x00. Since the predetermined address of the slave matches the address field in the message, the first slave in the line responds by echoing the message back to the master, over port P_1. The master detects an echo, and calculates the total number of detected devices by incrementing an address field by one. Now the master repeats sending a test message, this time with the address field 0x01. The first node in the line will not respond to the test message, since its predetermined address is 0x00. Because there is no match, the node will decrement only the address field, and pass the message to the next node in the line through the P_2 port. After passing the message, the node will change the direction of information flow, by toggling the values of DIR_1 and DIR_2 pins, thus configuring the P_1 port as a driver and P_2 port as receiver. The second node receives a message on P_1 port, now with the address 0x00, and responds to it by sending the echo, using the same port and address 0x00. The preceding node receives the echo on its P_2 port, increments the address field to 0x01, and passes the message to the master using P_1 port. Upon receiving an echo, the master increments the address field, updating the number of counted nodes to 0x02. If the total number of nodes in the line is given by n, the described process repeats for a total of n times. Successful addressing of the n-th node is presented in Fig. 10.

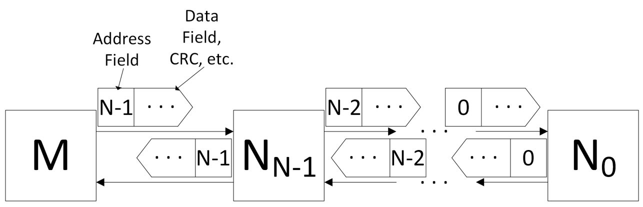


Fig. 10. Domino method addressing

Lastly, the master sends a test message once again, this time reaching the last node in the line, with the address of 0x01. The last node receives the message with this address field, decrements it and attempts to send it using its P_2 port to the next node. Since there are no nodes left, there will also be no echo message back to the master. The master waits for the echo for a certain amount of time, after which a timeout event occurs, signaling the master that there are no nodes left.

Once the total number of devices is determined, it can be used for communicating with each device. This communication protocol is based on master sending a message with the address field corresponding to the relative position of the node in the line.

One of the main issues with the described approach lies in addition of new nodes into a network. When the node is added, the initialization routine has to be repeated. This may be performed by the request from the user, or by periodical repetition of the algorithm. The suggested approach would be to manually start the routine, as it is less time-consuming.

The other downside of the proposed solution is the handling of a malfunctioned node or electrically corrupted bus - the line will be interrupted as the signal can not propagate through subsequent nodes.

Domino method occupies two serial ports of the node's micro-controller, which can be a limiting factor in system design, potentially increasing the overall cost of the system. Additionally, this solution introduces a significant computational delay, as the information is processed by each micro-controller it passes through. The total delay added into a single communication cycle, consisting of a single master request and slave response when addressing i-th slave in the line, is given by:

$$T_{DELAY_I} = 2 \cdot i \cdot T_{MCU}, \tag{1}$$

where T_{MCU} is the time required for a message to be processed and/or modified. This value has to be multiplied by the number of nodes that information passes through, and, since the message propagates back to the master, the whole value is multiplied by two. When evaluating T_{MCU}, it is important to include not only the time required for increment/decrement operations, but also time required for receiving and sending a modified message using a serial port. Assuming there are n nodes in the line, the maximum delay introduced by the Domino method is given by the expression:

$$T_{MAX} = 2 \cdot n \cdot T_{MCU}. \tag{2}$$

B. Staged Address Assignment with Bus Bridging (Pontoon Method)

As it can be observed from the previously described method, the core of the node constantly processes received information, introducing delay. This potential overload of the slave's micro-controller can be avoided by bridging the RS-485 bus inside a node using a discrete multiplexer (Fig. 11).

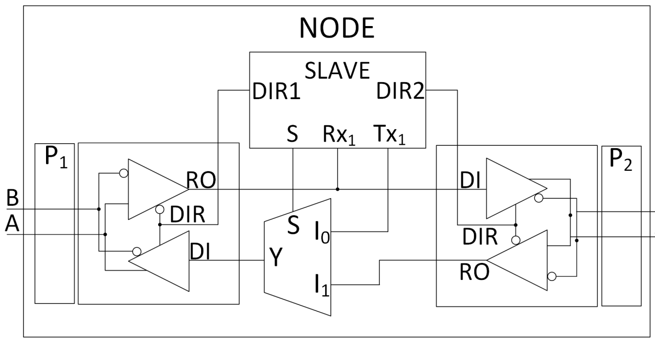


Fig. 11. Node structure for the Pontoon method

On a system level, the topology is identical to the one described in Fig. 9. Contrary to the previous approach, addresses of the nodes can be assigned by the master and each address has to be stored inside of the corresponding node. Additionally, only one serial port is used, which allows for more flexibility when designing a system. Initially, every node of the network is configured in a way that both of its ports receive the information, and each node's address is set to a predetermined value, e.g. 0x00. The master starts the process of address assignment by sending a message, addressing a device with the predetermined address. The first node receives the message, recognizing its address. Since its P_2 port is in receive mode, the propagation of the message stops. After recognizing its address, the first node sends an echo to the master by selecting I_0 multiplexer input and changing the direction of P_1 port, asking for a new address. The master sends another message for slave address configuration, where the address field can be arbitrarily determined by the user. Upon receiving a new address, the first node sends back a conformation message to the master and changes the direction of P_2 port in order to allow for the next address configuration message to pass through it. The master acknowledges that the first node is successfully configured and sends the message with the address 0x00 once again. The first node receives this message on P_1 port, and since its P_2 port is configured as a driver, the message can pass through the node, without unnecessary computing delay. As the information is passed, both ports of the first node have to change directions and I_1 multiplexer input has to be selected, to allow for opposite data flow. The second node in the line receives the message on its P_1 port. Because its address is 0x00 and its P_2 port is blocking the information flow, it echoes the message back to the master, requesting a new address. The first node receives an echo on its P_2 port, passes it through P_1 port and changes the direction of the ports once again. The process continues until the last node is reached, and once again, the master has to implement a waiting mechanism, when the last node tries to pass the configuration message. If a certain slave recognizes its address after the initialization is done, it has to configure its P_1 port as a driver, selecting the I_0 multiplexer input.

This time, adding new nodes to the network does not require repetition of the whole initialization process, but only a single passing of the configuration message. Again, the line will be interrupted in a situation where the node is malfunctioning or the bus is somehow corrupted, either by open or short circuit.

Contrary to the Domino method, this approach introduces additional delay during network configuration. For any node in a network, configurational delay is:

$$T_{CNFG_{SINGLE}} = 2 \cdot T_{MCU}, \tag{3}$$

where, again, T_{MCU} is the time required for a message to be received, processed and transmitted. It takes single T_{MCU} to process a message containing the predetermined address, and another T_{MCU} to process a message containing a new node address. For a network containing n nodes, the total configurational delay is given by:

$$T_{CNFG_{TOTAL}} = 2 \cdot n \cdot T_{MCU}. \tag{4}$$

After the network is successfully configured, the duration of a communication cycle, consisting of a message transmission from the master and a response from any slave is:

$$T_{COMM} = T_{MCU}, \tag{5}$$

since the message passes through preceding nodes directly, without being processed by their micro-controllers.

IV. Conclusion

From the hardware perspective, both automatic configuration methods presented require an additional RS-485 transceiver, which increases design complexity and overall price of each node. However, these approaches completely eliminate the need for manual work, except when adding or removing a node. By using the described algorithms for node communication, the whole network can be effortlessly reconfigured through master. This means that the nodes do not have to be easily reachable, which avoids implementing galvanic isolation of RS-485 transceivers, except when common mode voltage reduction is required. Further steps can be taken in order to completely relinquish the use of galvanic isolation [7].

The Domino approach that utilizes the network topology for slave addressing does not require individual storing of a slave's address. Therefore, it can be used in a system where there are multiple slaves of the same type, often performing a similar task. The example of a system that can benefit from this solution can be temperature monitoring networks, or street light management systems, where each node is functionally the same.

Additionally, this solution is less complex hardware-wise than the Pontoon method, but occupies an additional serial port and introduces computational delay, as the micro-controller processes address fields of the message.

Pontoon approach allows each node to have a unique address, independently of its position in a communication line. This can be useful in a system where certain types of sensors or devices have to share a specific address range, for example in various industrial or home automation systems.

Since there is no need for master to periodically poll the network in order to detect new devices, the whole method of network reconfiguration, adding and removing the nodes, is simplified compared to the first proposed method. This approach introduces virtually no additional communication delay, apart from the propagation of a signal through multiplexer. Pontoon approach does introduce a multiplexer in the node's design, but offers a possibility of compromise price-wise, since it occupies a single serial port, allowing for a simpler micro-controller. The usage of a discreet multiplexer can be avoided by directly connecting RO line from P_2 port to the DI line from P_1 port and Tx_1 line, and by keeping Tx_1 line in a high-impedance state whenever the message is expected to be received from P_2 port.

The Domino method does not require node address configuration, meaning that there is no time delay introduced during network initialization, compared to the Pontoon method that introduces a delay of $2nT_{MCU}$ during this phase, for a network consisting of n nodes. On the other hand, during a communication cycle, the Domino method requires $2iT_{MCU}$, for i-th node in the line, while the Pontoon approach requires only T_{MCU} for the same operation, regardless of the node position.

Both of the proposed solutions can be used in systems that are inherently linearly connected. In addition to the examples mentioned before, those systems include metallurgical furnaces, speed and traffic detection, environmental, and other kinds of roadway sensors, as well as various sewer sensors. Furthermore, the described methods can relatively easily be generalized in order to be applied to other communication systems, using different electrical standards and protocols. Without change in principal node structure, approaches can be easily adjusted to fit other serial standards. These methods of address assignments can be adapted for different communication mediums, including wireless and optical communication.

The future work should cover defining methodology for generalization of the presented address assignment approaches, as well as finding ways to improve system robustness and failure recovery capabilities.

Acknowledgments

This work was supported by the Ministry of Education, Science and Technological Development, Republic of Serbia, under Projects TR32043 and TR32039.

References

[1] *Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*, ANSI/TIA/EIA-485-A-1998, TIA/EIA, 1998.

[2] J. Axelson, *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*, 2nd ed. Madison, WI, USA: Lakeview Research LLC, 2007.

[3] T. Kugelstadt, "Application Report - The RS-485 Design Guide," TI, Dallas, TX, USA, 2021.

[4] Texas Instruments, "SN65HVD7x 3.3-V Full-Duplex RS-485 Transceivers with ±12-kV IEC ESD," TI, Dallas, TX, USA, 2019.

[5] REFLEX Thinking Solutions, "Busmodul - Profibus DP, Operating Manual," Reflex Winkelmann GmbH, Ahlen, Germany, 2014.

[6] C. Boiano, P. Guazzoni, L. Zetta, C. Guazzoni, and A. Pagano, "A 16-channel programmable antialiasing amplifier", IEEE Nuclear Science Symposuim & Medical Imaging Conference, Knoxville, TN, USA, 30 Oct. - 6 Nov. 2010.

[7] N. S. Jovičić and V. M. Rajović, "A Floating Linear Voltage Regulator for Powering Large-Scale Differential Communication Networks," *IEEE Access*, vol. 6, pp. 24669-24679, May, 2018.