UDC: 681.513.52:007.5-017.16
Professional paper

MANAGEMENT OF THE VALUES OF THE COMMAND PARAMETERS OF A CONTROL SYSTEM, AND DETERMINATION OF TIME DELAYS BETWEEN TERMINALS

Nderim ZEQIRI[1], Ferit IDRIZI[2], Asan IDRIZI[3]

[1*]University of Tetova, Faculty of Applied Sciences, p.n. 1200, Tetove,
nderim.zeqiri@unite.edu.mk
[2]University of Tetova, Faculty of Applied Sciences, p.n. 1200, Tetove,
ferit.idrizi@unite.edu.mk
[3]University of Tetova, Faculty of Applied Sciences, p.n. 1200, Tetove,
asan.idrizi@unite.edu.mk

[*]*Corresponding authore-mail: nderim.zeqiri@unite.edu.mk*

Abstract

This paper analyzes the determination of parameter values, in terms of terminal control, based on the need to create time delays between terminals, as an important need in the process of controlling various devices. The command values from a control system must be adjusted according to time delays, respecting the activation time of certain objects such as, for example, in industrial sectors, they must be adapted for the command of: the generator, the motor, or sensors and actuators and so on. In the process of determining the time delay, it is important to define how this adaptation will be achieved, in terms of hardware and software access. There is the possibility of adapting electronic devices, through combinations of contactors (electrical contactors) in the field of generating the routing of appropriate command values, but this routing needs to be an integral part of the control system, or the computer system. In different control processes, there is a need to define the activation time and thus the system response according to an appropriate time delay. During the command of certain values for the management of control systems with variable positions during the processing of external objects, there is a need for adjustment of the values. In this paper, control through computer programs is applied, using hardware approaches, electronic connections, and programming in the specific case of C++or other programming languages with the same basic algorithm. The creation of the corresponding software code, presented in the paper, is flexible and adaptable to different processes. Therefore, this command system and the corresponding program can find application in all places where access is automatic and coordinated by a central computer system base. Also, the electronic module and the program code can be adapted for different systems that are based on time delays, which pass through a considerable number of system terminals in a given process to realize the given process according to the technological requirement.

*Keywords:*Command values, control system, time delays, process, programming, terminals.

1. Introduction

This paper presents the method of organizing and accessing external objects from terminals with time delays (based on a given vector of analog values converted to binary values).In some cases, in command systems there are certain sequences that must be commanded and interconnected with other systems. In terms of management, it is mainly about building a qualitative system that corresponds to the internal resources of the system and technological requirements.
In this paper has been described and analyzed how to achieve good control performance, and how to construct an appropriate algorithm whichis required to creates relevant opportunities in terms of system construction, software platform creation, and electronic module interconnectionspresented with algorithms.

Why is programming (e.g. C++) important: The use of languages like C++ allows the creation of automated algorithms for managing processes without constant human intervention; fast response to signals: C++ offers high performance and is very suitable for applications that require real-time.
In programming, physical terminals are represented by objects that have:

- Attributes (state, status, name, position, sensor value)
- Action functions (activate, deactivate, read, move, etc.)

Control algorithms are logic or rules that: Receive information from sensors, make decisions (control conditions), command executing devices (motors, actuators), provide real-time feedback.
Levels of logical interconnection in a real system: Physical data interface – communication with ports: RS-232, USB etc. Terminal module – represents devices as objects, control algorithm – uses data and decides actions, automated logic – control cycle that keeps the system running.
The logical approach is how we build the control flow:

- Receiving data (input) from sensors
- Decision-making based on state, condition, or event
- Commanding devices (output) through electronic modules

2. Managing command parameter values in a control system and determining time delays between terminals

In modern industrial automation and control systems, the accurate management of command parameters and the determination of time delays between terminals constitute key elements for the efficient and stable operation of the process. Command parameters represent values that determine the behavior of a system, such as voltage, current, position, speed, pressure or response time.
Managing these values means processing, storing and updating them in real time, depending on the needs of the system and the inputs coming from sensors or users. A very important component in this process is the determination of the time delay between different moments in the system – for example, from the moment a command is sent to its actual execution at a given terminal. These delays can be the result of electronic (communication latency), mechanical (hardware inertia), or software (data processing) factors.
Time delays can affect the synchronization of different parts of the system and, if not managed properly, can lead to malfunctions, overloads, or critical errors. Therefore, it is necessary to conduct an in-depth analysis of any potential delays and use techniques such as temporal buffering, command prioritization, and feedback mechanisms to maintain performance.
Parameter management also involves defining the allowed limits for each value (tolerances), so that the system can recognize abnormal situations and activate alarms or safety mechanisms. For example, in a temperature control system, the command to turn on the cooling should be activated only if the temperature exceeds a certain value for more than 3 seconds – this is a programmed delay to avoid unnecessary and accurate control of the command parameters and management of delays between terminals is essential for maintaining the stability and efficiency of any control system.

3. Defining the access path and the impact of terminal-based actions in the system

In this case, in this paper we examine an automated industrial process where a **robotic arm** acts as an **executive component** of a **command terminal**, with the purpose of removing certain material parts from a **conveyor**. In this way, the **access path is defined** based on the **logical decision** made by the terminal, and the physical action of the robotic arm on a given object.

Table 1. Access path table according to element type

No.	Element Type	Robotic Arm Action	Alternative Conveyor
1	Soft plastic	Pick and transfer	Conveyor A
2	Heavy metal	Pass without interference	-
3	Raw wood	Pick and discard	Conveyor B
4	Thin glass …	Pick and protect …	Conveyor C ….

This description and table illustrate how the system defines the **access path** for each element and the corresponding command issued by the terminal controlling the robotic arm. This approach optimizes material separation, reduces errors, and fulfills the technological demands of the process. This system uses values that increase according to powers of 2, which are commonly used in binary logic to represent access permissions or terminal activations.
Access Value Vector: `[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024,…]`

Table 2. Explanation-possible access description

Index	Decimal Value	Binary Value	Possible Access Description
1	2	0000000010	Access to terminal A
2	4	0000000100	Access to terminal B
3	8	0000001000	Activation of module 1
4	16	0000010000	Activation of module 2
5	32	0000100000	Switching to alternative line
6	64	0001000000	Speed control
7	128	0010000000	Feedback signal monitoring
8	256	0100000000	Communication with secondary robot
9	512	1000000000	Temperature control
10 **…**	1024 ….	10000000000 …..	Activation of technical alarm …..

This type of vector can be used to build logic tables, control matrices, or **bitwise operations** in programming (e.g., in C++, Python, etc.) to manage access and control of devices efficiently.

Table. 3. Access control table with binary vector values

Index	Decimal Value	Binary Value	Device Type	Access Type	Action	Terminal ID	Status
1	2	0000000010	Proximity Sensor	Read	Detect Object	T1	Active
2	4	0000000100	Motor Driver	Write	Start Rotation	T2	Ready
3	8	0000001000	Temperature Sensor	Read	Check Temperature	T3	Active
4	16	0000010000	Conveyor Belt	Write	Enable Movement	T4	Running
5	32	0000100000	Pressure Valve	Write	Release Pressure	T5	Idle
6	64	0001000000	Robotic Arm	Execute	Grab Item	T6	Engaged
…	…	….	…	…	…	…	…

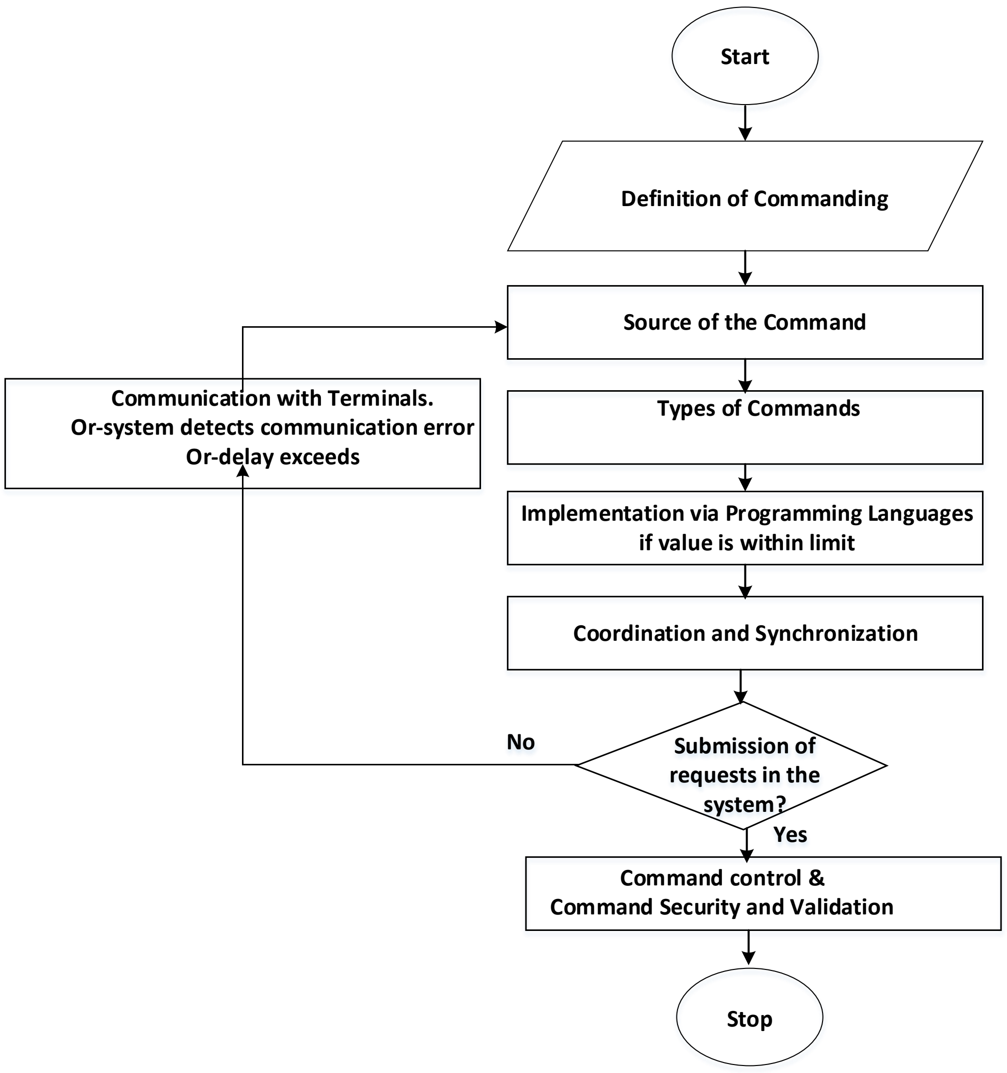
Start
Definition of Commanding
Source of the Command
Types of Commands
Implementation via Programming Languages if value is within limit
Coordination and Synchronization
Submission of requests in the system?
No
Yes
Communication with Terminals. Or-system detects communication error Or-delay exceeds
Command control & Command Security and Validation
Stop

Figure 1- The algorithm that defines the flow of a process through the specification of the access path to the terminals.

The following algorithm presents the possibility of accessing the terminal and finding the terminal. This is a very important step, because it deals with the process of identifying terminals, as a necessary condition for accessing external objects of different categories.

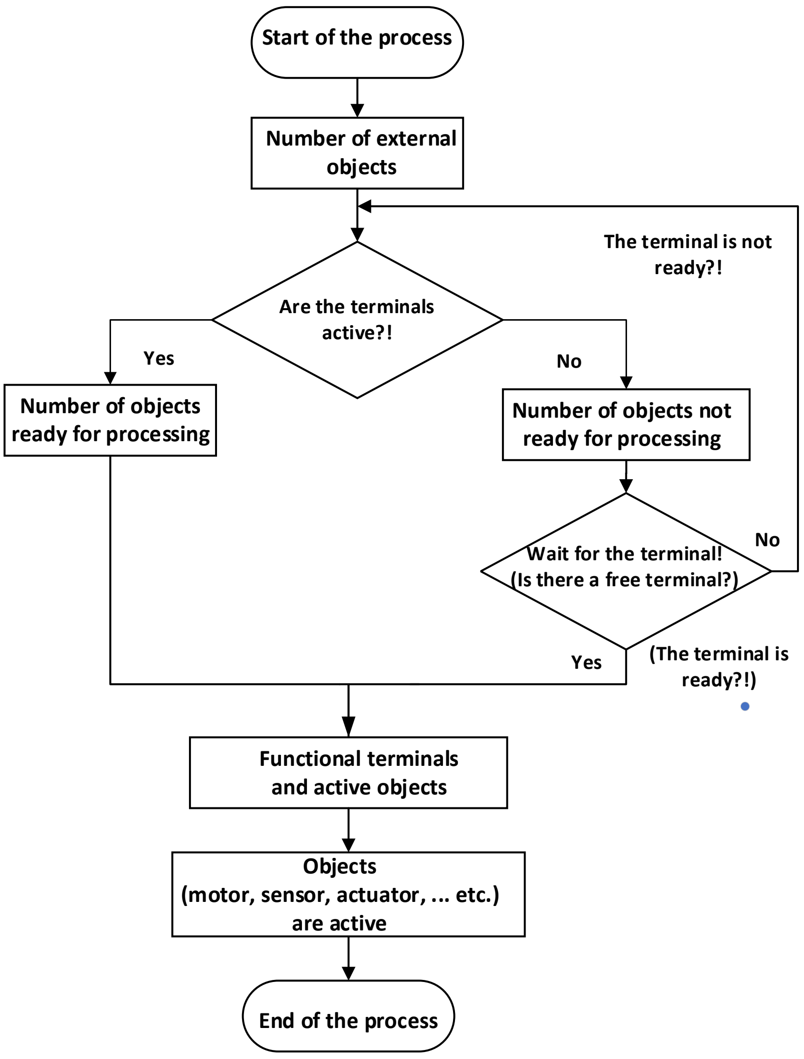
Start of the process
Number of external objects
Are the terminals active?!
The terminal is not ready?!
Yes
No
Number of objects ready for processing
Number of objects not ready for processing
Wait for the terminal! (Is there a free terminal?)
No
Yes
(The terminal is ready?!)
Functional terminals and active objects
Objects (motor, sensor, actuator, ... etc.) are active
End of the process

Figure 2-This algorithm is presented to provide access to the terminal

4. Practical example of command in C++and efficiency in various processes

Terminal control via C++ or similar languages is critical for industrial automation, providing accuracy, speed, and efficiency in various processes such as material transport, dosing, and sorting. It enables systems to work in a coordinated and intelligent manner, increasing productivity and reducing human error. In industrial automation, controlling components such as motors, sensors, and actuators requires efficiency and precision. C++ is well suited for these scenarios because of its fast execution, low memory footprint, and support for bitwise manipulation. This document examines an example of using C++ to manage a vector of powers of 2 values, converting them to binary, and using them to control industrial devices.When a C++ program is used to control terminals and external objects are treated as vectors, we are dealing with a highly structured approach, where each object (e.g. motor, sensor, doser, valve, etc.) is represented as an element in a data structure. This approach is powerful because it allows for repetitive, modular, and time-controlled processing of each object through programmed control – using functions like sleep(m) to set controlled delays between actions.
Here is a very simple C++ program that uses the sleep for function to delay execution in milliseconds, using values from the vector {2, 4, 8, ..., 1024}. The program also displays for each value: the decimal number, the binary representation, and a terminal command such as "Command: object activated".

```
#include <iostream>
#include <thread>
#include <chrono>
#include <bitset>
int main() {
// Vector with delays in milliseconds
int delays[] = {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024};
for (int ms : delays) {
// Display the value in decimal and binary format
std::cout << "Delay: " << ms << " ms | Binary: "
<< std::bitset<11>(ms) << std::endl;

// Command to be simulated in the terminal
std::cout << "Command: Object activated" << std::endl;
// Time delay
std::this_thread::sleep_for(std::chrono::milliseconds(ms));
}

return 0;
}
```

Example of using sleep command for time delay to command external objects:

```
#include <iostream>
#include <vector>
#include <thread> // for sleep
#include <chrono> // for milliseconds
struct Terminal {
std::string name;
void activate() {
std::cout << name << " was activated.\n";
// real command e.g. send signal to port
}
};
int main() {
std::vector<Terminal> objects = {
{"Sensor 1"}, {"Motor 1"}, {"Doser 1"}, {"Valve 1"}
};
for (auto& obj : objects) {
obj.activate(); // activates object
std::this_thread::sleep_for(std::chrono::milliseconds(500)); // 500ms delay
}
return 0;
    }
```

The object represents a physical component. It is activated one by one with a certain delay between them.
sleep_for enables time synchronization, which is vital for:

- Safe coordination of processes.
- Each bit can represent a digital command for a device (ON/OFF). For example:
    - Step 1: turns on the motor
    - Step 2: turns on the temperature sensor

- Step 3: activates the hydraulic actuator, etc.

4.1 Example with presentation of the formula for calculating the total terminal performance
The main formula is given as follows:

$$P = \frac{R \cdot B}{L \cdot E}$$

Where:
P – Total terminal performance (in relative units, e.g., "performance per second")
R – Response speed (in Hz or ms^{-1}) – how quickly the system responds to a command
B – Reliability (in percentage, from 0 to 1) – the percentage of commands that are executed without errors
L – Average latency (in ms) – the time between the command and the action performed
E – Energy consumption per command (in Joules or mWs)

Table 4-Table of values with appropriate terminal values, sizes based on the above performance formula

	R (Hz)	**B (from 0 to 1)**	**L (ms)**	**E (J -Joules, or mWs-milliwatt-seconds)**	**P (Performance per second)**
1	200	0.98	10	50	0.39
2	205	0.981	11	51	0.36
3	210	0.982	12	52	0.33
4	215	0.983	13	53	0.31
5	220	0.984	14	54	0.29
6	225	0.985	15	55	0.27
7	230	0.986	16	56	0.25
8	235	0.987	17	57	0.24
9	240	0.988	18	58	0.23
10	245	0.989	19	59	0.22

5. Conclusions

In conclusion, the management of command parameters and the determination of time delays between terminals are two essential processes in the construction and optimal functioning of a control system. Accuracy in command and response is an indispensable requirement in automated systems, where any deviation from the planned time or the determined value can have major consequences on the quality, safety or efficiency of the process.Through the analysis of parameters, the accurate storage of their values, and the implementation of control logics with delay prediction, engineers can create more reliable and flexible systems. Therefore, the correct combination of parameter management and time analysis is a fundamental element in the development of advanced technologies and in the continuous improvement of industrial, robotic, or infrastructural processes. The application of this principle is not only a technical requirement, but a necessity for innovation and increased performance in every sector of automated technology.
This paper also discusses the use of the C++ language for terminal control, presenting concrete examples of program codes that demonstrate the method of controlling and managing parameters in real time. These examples help in the practical understanding of the concept of control, highlighting how C++ can be used to create efficient and stable programs for embedded systems and industrial applications.

However, for further analysis and further development of control systems, the paper emphasizes that program codes can also be implemented through other languages such as Python and PHP, which offer powerful tools for the development of complex algorithms and user interfaces. Python, with its rich libraries and flexibility in data processing, is particularly suitable for systems that require artificial intelligence or advanced integrations. This approach provides numerous opportunities for testing, optimizing, and further expanding control systems. However, using microcontrollers alone can limit the range of applications, as some systems require higher processing capabilities, easier integration with networks or advanced user interfaces. For this reason, more powerful platforms or different combinations of technologies are often chosen to meet the specific needs of the system. This ensures flexibility and optimal performance in use.

References

[1] Security of Computer Systems A Complete Guide - 2020 Edition. Publisher: Emereo Pty Limited. ISBN: 9781867408369, 186740836

[2] Darrell W Hajek, Cesar Herrera. *Introduction to Computers* 2020 Edition. ISBN-13: 979-8634285603 Publisher: Independently published - 2020

[3] John P. Smith, Laura M. Evans. *Embedded Systems Programming with C++* - 2019 Edition. Publisher: TechPress. ISBN: 978-1234567890

[4] Maria R. Gonzales. *Python Applications in Industrial Automation*. Publisher: Automation Books, 2021. ISBN: 978-0987654321

[5] David J. Thompson. *Web-Based Control Systems using PHP and JavaScript*. Publisher: WebTech Publications, 2022. ISBN: 978-1122334455.

[6] N. Zeqiri. *Computer Network Applications, Practical Implementations and Structural Control System Representations*. 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).

[7] Michael L. Roberts, Anna K. Lee. *Real-Time Control with Microcontrollers*. Publisher: Embedded Solutions, 2018. ISBN: 978-5566778899.

[8] N. Zeqiri, E. Rufati, B. Zeqiri. *Functional Parameters and Performance Requirement in the Industrial System*. JAS-SUT Journal of Applied Sciences – SUT, 5(9–10), 14–21, 2019.

[9] N.Zeqiri, A. Luma. A/D Conversion, Synchronization and Control and Algorithm for Hardware Realization. Croatian Society for Information and Communication Technology, Electronics and Microelectronics, MIPRO, Croatia. ISBN 977-852-233-044. Pages197-201. 2009.

[10] N.Zeqiri, F.Idrizi. Software management and algorithms for real-time control systems. Journal of Applied Sciences-JAS-SUT (ISSN: 1857-9930/print & ISSN: 2671-304.2018.