The object of this study is modern Human-Machine Interfaces (HMI) and SCADA systems in the industry. The subject of research is techniques for optimizing the number of tags (variables) in the SCADA/HMI environment to enhance resource utilization efficiency.

One of the challenges in creating SCADA/HMI-based solutions can be the number of tags (variables) in the runtime environment. A large number of tags can lead to a problem of limited available resources.

The technique presented here allow for the optimization of the number of tags used in Human-Machine Interface systems built with SCADA software and operator panels in combination with Programmable Logic Controllers (PLCs).

An evaluation of the efficiency of techniques for reducing the number of HMI tags was conducted on an experimental configuration consisting of objects such as discrete input/output, analog input/output, actuators such as valves with discrete/analog control, and drives with frequency converters. The optimization coefficient, defined as the ratio of the number of input/output tags used directly to the number of tags after applying the optimization principle, was used as the efficiency criterion. Depending on the techniques and their combinations, the criterion values reached orders of 4, 10, and in one case even more than 100. These values are explained by the application of multiplexing approaches and various packing techniques.

The advantages and disadvantages of the reported techniques, as well as their application limitations, have been identified. Some techniques are suitable only for specific tasks.

These techniques could be applied in practical implementation when designing modern high-efficiency Human-Machine Interfaces under conditions of limited resources

Keywords: human machine interface, tag, resource optimization, tag licensing

UDC 004.5:004
DOI: 10.15587/1729-4061.2024.309029

DETERMINING THE EFFICIENCY OF TECHNIQUES FOR OPTIMIZING THE NUMBER OF TAGS IN MODERN HUMAN-MACHINE INTERFACES UNDER CONDITIONS OF LIMITED RESOURCES

Volodymyr Polupan
Corresponding author
PhD*
E-mail: polupanvv@nuft.edu.ua
Roman Mirkevych
PhD*
Oleksandr Pupena
PhD*
Oleh Klymenko
PhD*
Oleksii Mirkevych
PhD Student*
*Department of Automation and Computer Technologies of Control Systems named after Prof. A. P. Ladanyuk
National University of Food Technologies
Volodymyrska str., 68, Kyiv, Ukraine, 01601

Received date 01.05.2024
Accepted date 17.07.2024
Published date 30.08.2024

***How to Cite**: Polupan, V., Mirkevych, R., Pupena, O., ,Klymenko O., Mirkevych, O. (2024). Determining the efficiency of techniques for optimizing the number of tags in modern human-machine interfaces under conditions of limited resources. Eastern-European Journal of Enterprise Technologies, 4 (2 (130)), 52–66. https://doi.org/10.15587/1729-4061.2024.309029*

1. Introduction

Modern control and monitoring systems, as well as human-machine interfaces (HMIs), are extremely important for automated processes in various industries, including manufacturing, energy, transportation, construction, and infrastructure. HMI systems provide operators with the ability to interact with equipment and control its operation. However, it is important to understand that the resources available for use in an HMI system are not always limitless. Such resources include the amount of memory, computing power, bandwidth of communication channels, availability of interfaces, and others. One way or another, most of these resources are related to the number of real-time database variables, which are called different concepts in different systems. Hereinafter, the term "input/output tag" or simply "tag" means a certain object that is associated with actual values. Also, the "tag" has such properties as time stamp, quality, and others, according to its external data source, for example PLC [1]. Therefore, a limit on the number of tags that can be used may arise due to limited computing or other resources of devices implementing HMI. An example of such a situation can be operator panels, which usually have a number of tags clearly defined by the manufacturer that can be used in the system.

In the case of PC-based HMI implementation using SCADA programs, as a rule, there are no problems with computing resources. However, there may be a question of the existing license of the SCADA program, or communication limitations of the device itself, which are not able to pass the required number of tags per unit of time. As a rule, most modern SCADA programs are licensed based on the number of tags used. Therefore, a situation may arise that the predicted number of tags, on the basis of which the necessary license was chosen, will differ from the actual one, which can significantly affect the project budget.

Therefore, the effective use of tags becomes one of the key factors for optimizing the operation of HMI systems.

To optimize the number of input/output tags, SCADA project developers use various techniques.

Thus, the study of the effectiveness of ways to optimize the number of tags in modern human-machine interfaces under conditions of limited resources is a very relevant topic, and scientific research on this topic is important. In particular, this is due to the fact that they are aimed at reducing the load on the limited resources of the control system (computing power, bandwidth of the control network), improving the accuracy of data, and reducing the period of their update. The results of such studies are needed in practice because they contribute to improving the quality of functioning of industrial control systems.

2. Literature review and problem statement

The relationship between tag optimization in SCADA and PLC systems and ISA (International Society of Automation) standards can be seen through their impact on automation system design approaches. Much useful information and procedures for the development of modern human-machine interfaces can be obtained from the popular and useful standards ISA-101, ISA-18.2, ISA-88, 95, 106, and their IEC counterparts. By using these standards when designing an automation system, one can achieve more efficient functioning and a better level of optimization.

ISA-101 (Human Machine Interfaces for Process Automation Systems) provides guidelines for the design of human-machine interfaces for industrial control systems. One of the aspects of the standard is the provision of effective ergonomics and interface comprehensibility for operators, as indicated by the results of study [2].

Work [3] reports the results of research into high-performance human-machine interfaces. They showed that the design of modern human-machine interfaces requires the integration of a large amount of data, especially for configuration displays, which are a key element of high-performance HMIs. This is necessary to solve the problem of insufficient formalization of information for model-oriented approaches. But the issues related to the optimization of the exchange and the shortcomings that arise when the system is saturated with an excessive number of tags remained unresolved. The reason for this may be the authors' use of powerful SCADA tools that have sufficient computing and communication resources, and the authors' lack of resources for the implementation of the tasks. However, the use of powerful SCADA tools mostly leads to an increase in the cost of the control system and the installation of excess capacity in the project. An option to overcome the relevant difficulties can be the use of less powerful HMI devices, which do not add excess power to the project and thereby reduce the cost of project implementation. This is the approach used in work [4], in which the authors, scientists and engineers, determined that optimizing the number of tags affects the clarity and convenience of the interface and allows the use of simpler HMI devices. Also, in work [5] it was determined that it is necessary to comply with certain requirements regarding the design principles of modern HMIs. These include requirements for ergonomics, adaptability, and user orientation of the system, and to achieve these requirements, it is necessary to properly organize the work of the real-time database.

The authors of study [6] faced a similar problem. They determined that the introduction of new technologies within the framework of Industry 4.0 led to a significant increase in the volume of data in HMI systems. This increase, together with the limited cognitive capabilities of control system operators, complicates their interpretation. The solution to this problem is proposed by optimizing the representation of information to the operator as a means of increasing the operator's situational awareness.

ISA-18.2 (Management of Alarm Systems for the Process Industries) focuses on the management of alarm systems [7]. According to study [8], optimization of tags helps improve the detection and management of abnormal situations and, accordingly, affects the effectiveness of the signaling system. The authors of study [9] determined that effective management of the alarm subsystem involves the implementation of the necessary monitoring and configuration system. which leads to an increase in the amount of information in the control system. A similar conclusion was reached in study [10]; however, the authors did not emphasize that this, accordingly, leads to an increase in the number of HMI tags that must be constantly exchanged in the control system, which significantly increases the intensity of communication exchange.

ISA-88 (Batch Control) and ISA-95 (Enterprise-Control System Integration) standardize the control of batch processes and the integration of control systems. According to study [11], tag optimization contributes to data standardization and facilitates integration between different levels of automation. This is confirmed by the authors' research [12], in which practical examples show techniques for implementing the elements of the ISA-88 and ISA-95 standards. Paper [13] also provides an example of the development of application software for PLC and SCADA/HMI of high and medium algorithmic complexity. It gives examples of the need for optimal organization of the real-time database, but not enough attention is paid to it.

However, the common link between tag optimization in SCADA and PLC systems and ISA standards lies in their shared focus on improving the efficiency, reliability, and safety of production processes. By standardizing, optimizing resources and improving user experience. This allows enterprises to achieve better results in the management and monitoring of production processes. All this gives reason to assert that it is appropriate to conduct a study on the effectiveness of ways to optimize the number of tags in modern human-machine interfaces under conditions of limited resources.

3. The aim and objectives of the study

The purpose of our work is to establish the effectiveness of ways to reduce the number of tags used in the implementation of human-machine interfaces built on the basis of SCADA programs or operator panels in combination with a programmable logic controller (PLC). This will make it possible to implement a highly functional human-machine interface under conditions of limited available resources.

To achieve the goal, the following tasks were set:

– to systematize tags according to their purpose;

– to consider typical data structures used in PLCs to control various types of objects: discrete and analog PLC input/output variables, valve with discrete OPEN/CLOSE type control, valve with analog control, drive with frequency converter;

– to investigate the effectiveness of techniques that will make it possible to reduce the number of tags in a conditional object of automation based on typical structures, to determine the optimization coefficient of each technique, and to give an assessment of the possibility of using each of the techniques during the practical implementation of real systems.

4. The study materials and methods

The object of our research is modern human-machine interfaces (HMIs) and SCADA systems in the industry.

The main hypothesis of the study assumes that the use of certain optimization techniques will make it possible to significantly reduce the number of tags used, without significant loss of functionality.

The following assumptions were made in the research methodology:

– in the argumentation of licensing the number of tags as a limiting resource, it is assumed that it is the tags (variables) that count, and not their size in memory; there are SCADA programs that calculate limits in bit equivalent, they are not the subject of research;

– only external input/output tags are counted in the optimality criteria, i.e., those that directly participate in the exchange with PLC;

– structural tags are counted by the number of fields, as it is mostly accepted in SCADA programs.

In the study, the number of tags was considered a key indicator of resource limitations. This is a deliberate simplification adopted specifically for this study, which is typical when targeting relevant limitations in SCADA/HMI tools. This indicator is not relevant for its use as the main criterion for determining the resource intensity of the SCADA/HMI solution as a whole. To fully take into account the resource intensity of the solution, it is necessary to use a more complex criterion that takes into account the allocation of computing resources, I/O bandwidth, page update time, response time to the operator's actions, etc.

In the research, we used the following methods.

Analysis of typical control objects to determine the type of variables (tags) according to their purpose, their systematization and synthesis into ready-made structures. This method allows you to organize tags based on their functional purpose, which will help you use them to provide the desired functionality, and further understand exactly which variables are used in the system, how and when they are involved in the communication between the PLC and SCADA/HMI, and how they can be optimized to reduce the total number.

Analysis of typical data structures used in PLCs, such as discrete and analog PLC input/output variables, discrete and analog control valves, and frequency converter drives. This will help understand the main components of automated systems, which is necessary to identify potential areas for optimization.

Specifying a conditional automation object to determine the number of tags without optimization. This allows us to estimate the initial number of tags used in typical systems without applying optimization methods. This forms a basis for comparing results after implementation of optimizations.

Investigating the effectiveness of ways to reduce the number of tags through quantitative analysis and comparison of results. Calculation of indicators such as the optimization ratio (the ratio of the number of tags before and after applying the optimization technique) makes it possible to accurately measure how effective each technique for reducing the number of tags is. Using the calculated indicators, one can objectively compare different optimization techniques and draw conclusions about their effectiveness.

Assessment of practical implementation through implementation complexity analysis and practical tests. It is important to evaluate how the implementation of each optimization technique affects the overall complexity of the system and implementation, which includes an analysis of the complexity, time, and resources required for implementation, as well as possible problems that may arise during implementation. Testing optimization methods under real conditions makes it possible to obtain objective data about their effectiveness and identify possible problems.

Each of these methods contributes to the achievement of the main goal of the research – reducing the number of tags in human-machine interfaces, which will make it possible to optimize the use of resources and improve the efficiency of systems.

5. Results of investigating the effectiveness of ways to optimize the number of tags under conditions of limited resources

5. 1. Systematization of tags according to their purpose

The simplest way to implement an HMI is to display current values from sensors and buttons to control actuators. However, such an HMI will not satisfy even a complete list of basic functions. Let's consider this in more detail.

So, for example, in the ISA-101 standard, which directly concerns human-machine interfaces, as well as a number of sources on highly effective human-machine interface [2], for measurement parameters, it is recommended to use a context with which the operator is better situationally aware. That is, in addition to the current value of the measurement parameters, it is necessary to specify the state of the object. In the ISA-18.2 standard and its analog IES, HMI components must change their state (color, grayscale) depending on the state of the alarm associated with it. If alarms are generated in PLC, as is often the case when multiple HMIs need to be integrated with a single PLC, then the alarm states must be passed as part of the context. This all leads to the need for structural associations of variables responsible for values and states that indicate their context.

The situation with controls is even more complicated. In addition to the positional position of the controls, it is necessary to provide for the display and control of modes (manual/automatic, locked, etc.).

One way or another, to ensure the functionality required by modern SCADA/HMI tools in accordance with standards and best practices, the availability of status (contextual) information is mandatory. Whether this information will be combined into one structure with floating values, or whether it will be transmitted separately is a matter of convenience and compatibility.

The implementation of such functionality is provided by a number of standards and supported by many SCADA/HMI tools. Thus, in accordance with the standards ISA-88 (IEC-61512), ISA-95 (IEC-62264) and, more recently, ISA-106, during the design, development, and operation of software for production and technological process control systems, each object of automation is considered to be a separate entity. From a management point of view, the Equipment hierarchy stands out (see below), in which each object has its own role. In addition to Equipment, the ISA-95 (IEC-62264) standards also distinguish other resources within the enterprise, such as materials, personnel, and their combinations (process and product segments), as well as assets.

At the ASKTP level, according to the ISA-88 (IEC-61512) standard, all objects are divided separately into "technology" (how to make a product) and "Equipment" (what to use to make a product). The automation of "technology construction" applies only to productions with a variable recipe (the

ISA-88 standard was designed for this purpose). Unlike the technological part, Equipment automation applies to all types of production, even if only continuous processes with the same technology are used.

In all of these Equipment standards, functions and their relationships are aggregated into more general entities that are perceived as a whole. At the same time, concepts familiar to automation engineers such as "control device", "regulation circuit", "actuator", etc. become part of Equipment.

In any case, the approach to use in ISA-95, ISA-88, and ISA-106 Equipment and the use of contexts (states) have the same nature and are implemented in the same way.

For example, a 2-position valve or damper is given, as it is represented in a classic automation system:

– a control body (directly a valve or damper);

– actuator with one control pneumatic signal "OPEN";

– two end position sensors "OPEN", "CLOSED".

On the automation scheme (or P&ID) for each of the parts there will be one image (Fig. 1), which, as a rule, corresponds to one means of automation.

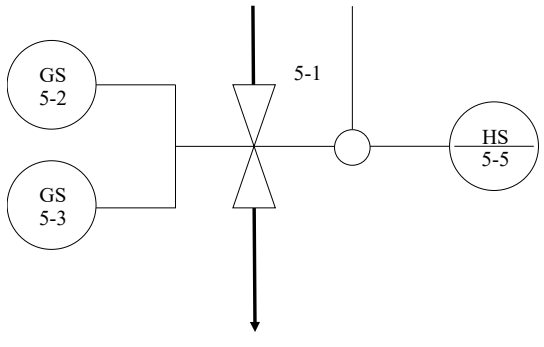

Fig. 1. Image of a valve on the automation scheme

In addition to this valve, there are functions that are often not shown on the automation diagram but must be provided in the PLC and SCADA/HMI algorithm, for example:

– basic control and management functions (for example, algorithm management);

– HMI interaction functions, such as manual/auto mode switching and manual control;

– alarm functions (for example, "not closed" alarm).

All these functions must be implemented in the controller and SCADA/HMI programs, so they will have the corresponding variables/tags and/or functions. In the classical representation, it can be separate variables and functions that are not logically combined into a separate entity.

From the point of view of operation, the valve is perceived not as a set of functions but in terms of its states, for example: functional ("open", "closed"), mode ("under manual mode", "under automatic mode"), alarm ("did not open "). These concepts are specific to the valve as a whole, not to its parts or functions. Therefore, on HMI displays, the automation tools can be shown as elements grouped together, the animation will involve the use of all the tags that are relevant for the valve. This concept is common to all the standards listed above.

Similarly, engineers who are not related to automation are related to it. They operate with concepts of states, not functions, or means of measurement and control. Therefore, during design or operation, they are not interested in the activation of end position sensors, and in the end, they may not be there at all, or only one of them may be present.

Thus, control over the Equipment is executed through the corresponding state variables [14], which must be in the program of the controller, SCADA/HMI, or other intelligent means. For discrete function states, these are bit states that take the values TRUE/FALSE, or a combination thereof.

In addition to measured values and variable states, modern HMI requirements include the possibility of setting, i.e., changing parameters. Examples of such parameters can be settings for scaling or filtering of analog input values, alarm settings, settings for regulator parameters, and much more. In the end, it turns out that the number of such parameters and, accordingly, tags for their implementation is several times more than the number of measurement values and control commands. However, they have one feature – they need to be changed and displayed only when necessary, which mostly happens often during debugging, but very rarely during system operation.

Thus, for the implementation of HMI, we can classify tags as follows:

– measurement, process control variables, and control commands that are required in real time;

– status, which are needed in real time;

– parametric, which are required upon request.

5. 2. Typical data structures

When implementing similar functionality in PLC, it is convenient for them to develop structures that will be processed as a single entity [15]. We shall assume that these structures implement data for Equipment. Consider typical data structures used in PLCs to control various types of objects, such as PLC Discrete Input/Output Variables, PLC Analog Input/Output Variables, OPEN/CLOSE Discrete Control Valve, Analog Control Valve, a drive with a frequency converter. The structures from PAC Framework [16] are taken as a basis as examples, but they may have a different appearance.

DIVAR is a data structure representing discrete process input variables (Table 1).

DOVAR is a data structure representing the discrete output variables of a process (Table 2).

Table 1

An example of a data structure used for a discrete PLC input variable

No.	ID	Data type	Description
1	STA_VRAW	bool	=1 – the value of the discrete signal
2	STA_VALB	bool	=1 – the value of the discrete input variable after all transformations
3	STA_BAD	bool	=1 – the data is unreliable
4	STA_ALM	bool	=1 – alarm
5	STA_ISALM	bool	=1 – involved as a technological alarm
6	STA_ISWRN	bool	=1 – involved as a technological warning
7	STA_WRN	bool	=1 – warning
8	STA_FRC	bool	=1 – boost mode
9	STA_SML	bool	=1 – variable in simulation mode
10	PRM_ISALM	bool	=1 – activate as an emergency alarm
11	PRM_ISWRN	bool	=1 – use as a warning alarm
12	PRM_INVERSE	bool	=1 – invert the raw value
13	PRM_NRMVAL	bool	value of the norm
14	PRM_QALENBL	bool	=1 – activate the unreliability alarm of the channel

AIVAR is a data structure representing the analog input variables of a process (Table 3).

AOVAR is a data structure representing the analog output variables of a process (Table 4).

Table 2

Example of a data structure used for a discrete PLC output variable

№	ID	Data type	Description
1	STA_VRAW	bool	=1 – the value of the discrete signal
2	STA_VALB	bool	=1 – the value of the discrete input variable after all transformations
3	STA_BAD	bool	=1 – The data is unreliable
4	STA_FRC	bool	=1 – Boost mode
5	STA_SML	bool	=1 – variable in simulation mode
6	PRM_INVERSE	bool	=1 – invert the raw value
7	PRM_QALENBL	bool	=1 – activate the unreliability alarm of the channel

Table 3

Example of a data structure used for a PLC analog input variable

No.	ID	Data type	Description
1	STA_BRK	bool	=1 – Channel break
2	STA_OVRLD	bool	=1 – Short circuit or channel congestion
3	STA_BAD	bool	=1 – The data is unreliable
4	STA_ALM	bool	=1 – active technology alarm and inactive BAD
5	STA_LOLO	bool	=1 – Critically low value
6	STA_LO	bool	=1 – Low value
7	STA_HI	bool	=1 – High value
8	STA_HIHI	bool	=1 – Critically high value
9	STA_WRN	bool	=1 – active technology warning and inactive BAD and ALM
10	STA_FRC	bool	=1 – Boost mode
11	STA_SML	bool	=1 – variable in simulation mode.
12	PRM_LOENBL	bool	=1 – LO alarm involved
13	PRM_HIENBL	bool	=1 – HI alarm is involved
14	PRM_LOLOENBL	bool	=1 – LOLO alarm is involved
15	PRM_HIHIENBL	bool	=1 – HIHI alarm is involved
16	PRM_BRKENBL	bool	=1 – a break alarm is involved
17	PRM_OVRLENBL	bool	=1 – overload alarm involved
18	PRM_PARAISPROC	bool	=1 – Setting parameters (hysteresis, non-sensory) are set as a percentage
19	LORAW	INT	Raw (unscaled) minimum value
20	HIRAW	INT	The raw (unscaled) value of the maximum
21	VAL	real	scalable value
22	VALFRC	real	maintains a forced value
23	LOENG	real	Engineered (scaled) minimum value
24	HIENG	real	Engineered (scaled) value of the maximum
25	LOSP	real	LO alarm setpoint
26	HISP	real	HI alarm setpoint
27	LOLOSP	real	LOLO alarm setpoint
28	HIHISP	real	HIHI alarm setpoint
29	THSP	real	HI technological setpoint
30	TLSP	real	LO technological setpoint
31	T_FLT	uint	filtration time in milliseconds (filter – aperiodic link)
32	VRAW	int	raw value
33	TDEALL	uint	Delay time for alarm LL to occur in 0.1 seconds
34	TDEAL	uint	Alarm delay time L in 0.1 seconds (if necessary)
35	TDEAH	uint	Alarm delay time H in 0.1 seconds (if necessary)
36	TDEAHH	uint	HH alarm delay time in 0.1 seconds (if necessary)
37	VALPROC	real	Percentage value

Table 4

An example of a data structure used for a PLC analog output variable

No.	ID	Data type	Description
1	STA_BAD	bool	=1 – The data is unreliable
2	STA_FRC	bool	=1 – Boost mode
3	STA_SML	bool	=1 – variable in simulation mode
4	VRAW	int	raw values
5	VAL	real	scalable value
6	VALFRC	real	retains the forced value
7	LORAW	int	Raw (unscaled) minimum value
8	HIRAW	int	The raw (unscaled) value of the maximum
9	LOENG	real	Engineered (scaled) minimum value
10	HIENG	real	Engineered (scaled) value of the maximum
11	VALPROC	real	percentage value

VLVD is a data structure representing a discrete-controlled valve of the "OPEN"/"CLOSED" type (Table 5).

Table 5

Example of a data structure used for an OPEN/CLOSE discrete control valve

No.	ID	Data type	Description
1	STA_IMSTPD	bool	=1 stopped in an intermediate state.
2	STA_OPNING	bool	=1 Opens
3	STA_CLSING	bool	=1 Closes
4	STA_OPND	bool	=1 Open
5	STA_CLSD	bool	=1 Closed
6	STA_DISP	bool	=1 remote mode (from PC/RAM)
7	STA_FRC	bool	=1 at least one of the variables in the object is forced
8	STA_SML	bool	=1 simulation mode
9	STA_BLCK	bool	=1 Blocked
10	ALMOPN	bool	=1 Did not open (Control circuit malfunction)
11	ALMCLS	bool	=1 Did not close (Control circuit malfunction)
12	ALMSHFT	bool	=1 Disturbances
13	ALM	bool	=1 VM Error
14	WRN	bool	=1 VM warnings
15	ALMPWR1	bool	= 1 – no power
16	ALMSTPBTN	bool	= 1 – stop button pressed
17	T_DEASP	uint	Alarm delay time of 0.1 seconds
18	T_OPNSP	uint	Maximum opening time in 0.1 seconds
19	CMD_OPN	bool	open
20	CMD_CLS	bool	close
21	CMD_TGL	bool	switch
22	CMD_ACK	bool	confirm the alarm
23	CMD_RESet	bool	reset alarms
24	CMD_MAN_AUTO	bool	switch manual/automatic
25	CMD_MAN	bool	enable manual mode
26	CMD _AUTO	bool	turn on automatic mode

VLVA is a data structure representing an analog-controlled valve (Table 6).

Table 6

An example of a data structure used for an analog control valve

No.	ID	Data type	Description
1	STA_OPND	bool	=1 Open
2	STA_CLSD	bool	=1 Closed
3	STA_DISP	bool	=1 remote mode (with PC/OP)
4	STA_FRC	bool	=1 at least one of the variables in the object is forced.
5	STA_SML	bool	=1 simulation mode
6	STA_BLCK	bool	=1 Blocked
7	POS	real	VM position (0–100%) – feedback
8	CPOS	real	VM position (0-100%) – setpoint
9	T_DEASP	uint	Alarm delay time of 0.1 seconds
10	CMD_MAN_AUTO	bool	switch manual/automatic
11	CMD_MAN	bool	enable manual mode
12	CMD _AUTO	bool	turn on automatic mode

DRV is a data structure representing a drive with a frequency converter (Table 7).

Table 7

Example of a data structure used for a variable frequency drive

No.	ID	Data type	Description
1	STA_MAINT	bool	=1 withdrawn from service.
2	STA_STOPING	bool	=1 Stops
3	STA_STRTING	bool	=1 Runs
4	STA_STOPED	bool	=1 Stopped
5	STA_ISREVERS	bool	=1 – start with reverse
6	STA_WRKED	bool	=1 Work in progress
7	STA_DISP	bool	=1 remote mode (with PC/OP)
8	STA_FRC	bool	=1 at least one of the variables in the object is forced
9	STA_SML	bool	=1 simulation mode
10	STA_BLCK	bool	=1 Blocked
11	ALMSTRT	bool	=1 Didn't turn on
12	ALMSTP	bool	=1 Didn't disconnect
13	ALMSHFT	bool	=1 Disturbances
14	ALMINVRTR	bool	=1 frequency converter error
15	ALMPWR	bool	=1 there is no power to the contactor
16	ALM	bool	=1 Drive error (by OR)
17	WRN	bool	=1 Drive warning (by OR)
18	SPD	real	VM speed/frequency (0–100 %) – feedback
19	CSPD	real	VM speed/frequency (0–100 %) is the setpoint
20	T_DEASP	uint	Alarm delay time of 0.1 seconds
21	CMD_STRT	bool	run
22	CMD_STOP	bool	stop
23	CMD_ ISREVERS	bool	Run with reverse
24	CMD_UP	bool	More
25	CMD_DWN	bool	Less
26	CMD_MAN_AUTO	bool	switch manual/automatic
27	CMD_MAN	bool	enable manual mode
28	CMD _AUTO	bool	turn on automatic mode
29	CMD _MAINTON	bool	decommission
30	CMD _MAINTOFF	bool	put into operation

As already mentioned, these structures contain measurement, status, and parametric information and for their implementation in SCADA, appropriate tags are required. At the same time, part of the status data may not be displayed on HMI as it has a certain redundancy and is needed only for the real-time operation of the PLC algorithms. However, when debugging, such information is also valuable, which will provide flexibility in setting up, configuring, and commissioning the system. As a counterweight to this statement, it can be pointed out that a large part of the data from the above structures does not have to be transferred from PLC to HMI, which will make it possible to significantly save the number of HMI tags. But at the same time, opportunities and flexibility will be significantly lost.

In a large number of SCADA programs, one bit variable (bool) and one variable of a larger dimensionally, both udint (32 bits) and string (up to 255 bytes), are considered equally in terms of the number of HMI tags. Thus, the main ways to optimize the number of HMI tags are to combine a certain number of tags into one HMI tag with a dimensionality larger than the original tags.

5. 3. Investigating the effectiveness of techniques to reduce the number of tags.

This study offers an analysis of techniques to optimize the number of used HMI tags without losing the functionality embedded in the above structure.

Generalized data on the number of HMI tags that will be used for this configuration of the automation object are given in Table 8.

Table 8

Summary table of used HMI tags for optimization

Type of data structure	Number of variables in the structure	Number of structures	Total number of HMI tags before optimization
DIVAR	14	700	9,800
DOVAR	7	400	2,800
AIVAR	37	100	3,700
AOVAR	11	20	220
VLVD	26	50	1,300
VLVA	12	20	240
DRV	30	30	900
Всього	–	–	18,960

A conditional automation object was used as the initial data for the study, which includes the following elements:

– discrete input variables (DIVAR) – 700 pcs.;
– discrete output variables (DOVAR) – 400 pcs.;
– analog input variables (AIVAR) – 100 pcs.;
– analog output variables (AOVAR) – 20 pcs.;
– valve with discrete control "OPEN"/"CLOSED" – 50 pcs.;
– valves with analog control – 20 pcs.;
– drive with frequency converter – 30 pcs.

5. 3. 1. Combining bit tags (bool) into double unsigned variables (UDINT/ULONG/DWORD)

One of the most common techniques to optimize the number of HMI tags is to combine a certain number of Boolean tags (usually up to 32) into one HMI tag with a size of 16 or 32 bits, for example, udint. It is fair to note that some systems have support for specific 64-bit data types, such as ULINT for S7-1500 PLCs. But since these types of data are not very common among most PLC/SCADA links today, they were not considered in this study.

The essence of the technique boils down to the fact that on the part of PLC, all bit variables that need to be transferred to SCADA must be pre-packaged, for example, into 32-bit variables. To this end, one can use the basic bitwise OR operation.

As a result of such manipulations, in theory, instead of 32 HMI tags of the bool type, from the point of view of SCADA, we shall receive only one HMI tag, in which each individual bit will determine the state of the corresponding variable in PLC.

From the SCADA side of the program, various techniques can be used to perform the reverse operation of "unpacking" a 32-bit variable. In many SCADA programs, there are built-in functions that allow one to immediately refer to a separate bit in one or another HMI tag. If there are no such functions, then one can use the bit AND bitwise operation with the appropriate mask.

If one analyzes the data structures, the following transformations can be performed in order to apply the specified technique to them.

All bit variables responsible for the state in the data structure should be combined into one 32-bit state variable (STA), and parametric bit variables responsible for the configuration of a particular object should be combined into one 32-bit parameter variable (PRM).

On the example of the DIVAR data structure – a data structure representing discrete process input variables, one can see that all the variables included in this structure have the bool data type. After applying the transformations proposed by this optimization technique, the DIVAR data structure will have the updated form given in Table 9.

Table 9

Representation of the DIVAR structure after concatenation of bit variables

No.	ID	Data type	Description
1	STA	UDINT	State bits
2	PRM	UDINT	Parametric bits

Thus, in terms of the number of HMI tags used, the non-optimized data structure used 14 HMI tags, and the one optimized in this way used 2 HMI tags.

It can be noted that 14 bit variables can be easily "packed" into one 32-bit HMI tag, but at this stage of the study we separate the bits responsible for the state and the parametric bits for ease of use when designing HMI. We shall return to this issue in the following optimization techniques.

The structure responsible for the actuator is considered separately. DRV is a data structure representing a drive with a frequency converter. In this data structure, one can also easily distinguish the bit variables responsible for the state of the actuator. We combine these variables into one 32-bit UDint type STA variable. Having analyzed this structure, we can conclude that this data structure does not include bit variables responsible for configuration but there are bit variables responsible for controlling this mechanism. Therefore, it is advisable to "pack" such variables into one 32-bit CMD variable. After all, assuming that only one command is transmitted at a time, it can be transmitted in a number that increases the number of commands to the bit size of the CMD word to the power of 2. Also, further optimization can lead to combining CMD and STA into a single structure, for example CSTA, but this may be more difficult to implement.

After applying these transformations, the data structure of the DRV type will have the form given in Table 10.

Table 10

Representation of the DRV structure after merging the bit variables

No.	ID	Data type	Description
1	STA	UDINT	State bits
2	SPD	real	VM speed/frequency (0–100 %) – feedback
3	CSPD	real	VM speed/frequency (0–100 %) – setpoint
4	T_DEASP	uint	Alarm delay time of 0.1 seconds
5	CMD	UDINT	Control command bits

As a result, it can be seen that in terms of the number of HMI tags used, the non-optimized data structure used 30 HMI tags, while the optimized one used 5 HMI tags.

Having optimized all the structures in this way, we obtained the results given in Table 11.

Table 11

The result of optimization by combining bit variables

Type of data structure	Number of variables in the structure	Number of structures	Total number of HMI tags after optimization
DIVAR	2	700	1,400
DOVAR	2	400	800
AIVAR	21	100	2,100
AOVAR	9	20	180
VLVD	2	50	100
VLVA	5	20	100
DRV	5	30	150
Total	–	–	4,830

A comparative diagram of the number of tags before and after optimization is shown in Fig. 2.

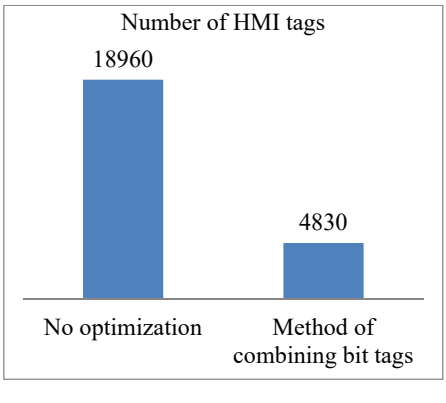

Fig. 2. Comparison of the number of tags before optimization and after using the technique for combining bit variables

The optimization coefficient for this technique is calculated according to the formula:

$$K_{opt} = \frac{N_{nopt}}{N_{opt}}, \quad (1)$$

where K_{opt} is the optimization factor; N_{nopt} is the number of HMI tags before optimization; N_{opt} – number of HMI tags after optimization.

For a given optimization technique:

$$K_{opt} = \frac{18,960}{4,830} = 3.93. \quad (2)$$

This approach is successfully used by many integrators.

5. 3. 2. Using the configuration buffer

As already noted, in addition to real-time data (values of variables, statuses), a large amount of parametric or so-called configuration data (CFG DATA) is associated with each data structure, which must be transferred to/from SCADA/HMI only when necessary.

Therefore, to reduce the large amount of configuration data circulating between the SCADA/HMI and the controller, it is suggested to use a buffer. For each array (set) of the same type of data structures or other objects, one can use a separate buffer.

To implement such an approach, each data structure must have a unique identifier within the set, by which it can be associated with a buffer. The structural diagram of the exchange of configuration data through the buffer is shown in Fig. 3.

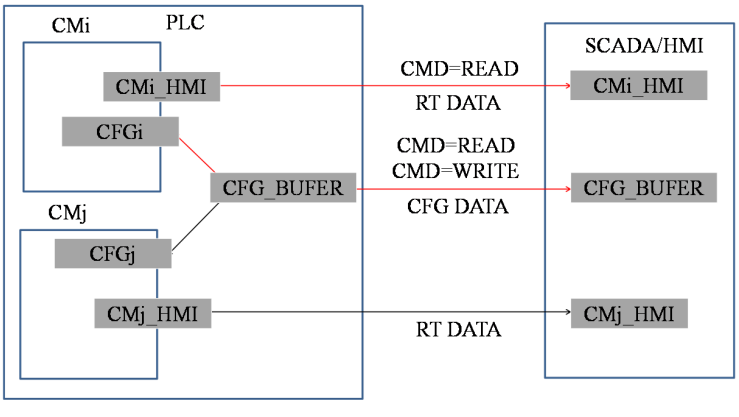

Fig. 3. Structural diagram of buffer configuration data exchange

The buffer structure variable from the SCADA/HMI point of view must contain the same fields as those structures that were given for the first technique of our study. The presence of a buffer makes it possible to abandon the use of configuration HMI tags in each of the data structures.

It is worth noting that the technique of optimization using a buffer is already based on the first technique described above and includes it.

Consider for example the AIVAR structure – a data structure representing analog input process variables. After optimization by the first technique, it takes the form given in Table 12.

Having analyzed this structure, we can conclude that real-time data includes the following elements of this structure: STA, VAL, VALPROC. The rest of the elements are configurational. Therefore, using a buffer, it is possible to optimize the AIVAR structure to the form given in Table 13.

Table 12

Representation of the AIVAR structure after optimization by technique 1

No.	ID	Data type	Description
1	STA	udint	Status bits
2	PRM	udint	Parametric bits
3	LORAW	INT	The raw (unprocessed) value of the minimum
4	HIRAW	INT	The raw (raw) value of the maximum
5	VAL	real	scaled value
6	VALFRC	real	stores the forced value
7	LOENG	real	Engineering (scaled) minimum value
8 HIENG real			Engineering (scaled) maximum value
9	LOSP	real	LO alarm setting
10	HISP	real	HI alarm setting
11	LOLOSP	real	LOLO alarm setting
12	HIHISP	real	HIHI alarm setting
13	THSP	real	Technological setting HI
14	TLSP	real	Technological setting LO
15	T_FLT	uint	filtering time in milliseconds (filter – aperiodic link)
16	VRAW	int	raw value
17	TDEALL	uint	The delay time for the occurrence of the LL alarm in 0.1 seconds
18	TDEAL	uint	Delay time for alarm occurrence L in 0.1 seconds (if necessary)
19	TDEAH	uint	Delay time for H alarm occurrence in 0.1 seconds (if necessary)
20	TDEAHH	uint	Delay time for HH alarm occurrence in 0.1 seconds (if necessary)
21	VALPROC	real	Value in percentage

Table 13

Representation of the AIVAR structure after real-time data separation

No.	ID	Data type	Description
1	STA	UDINT	Status bits
2	VAL	real	scaled value
3	VALPROC	real	Value in percentage

That is, it makes it possible to reduce the value of the number of used HMI tags from 21 to 3. But at the same time, there must be at least one buffer containing a complete set of data (Table 12).

It is worth noting that the data structure of the buffer may differ significantly from that given in Table 12, as it may be necessary in additional parameters, for example, the identifier of the current structure currently in the buffer.

After introducing a buffer to all data structures and their optimization, the general pattern regarding the number of used HMI tags takes the form given in Table 14.

A comparative diagram of the number of tags before and after optimization in this way is shown in Fig. 4.

Table 14

The result of optimization technique by adding a configuration buffer

Type of data structure	Number of variables in the structure	Number of structures	Total number of HMI tags after optimization
DIVAR	1	700	700
DOVAR	1	400	400
AIVAR	3	100	300
AOVAR	3	20	60
VLVD	2	50	100
VLVA	4	20	80
DRV	4	30	120
DIVAR_BUFER	2	1	2
DOVAR_BUFER	2	1	2
AIVAR_BUFER	21	1	21
AOVAR_BUFER	9	1	9
VLVD_BUFER	2	1	2
VLVA_BUFER	5	1	5
DRV_BUFER	5	1	5
Total			1806

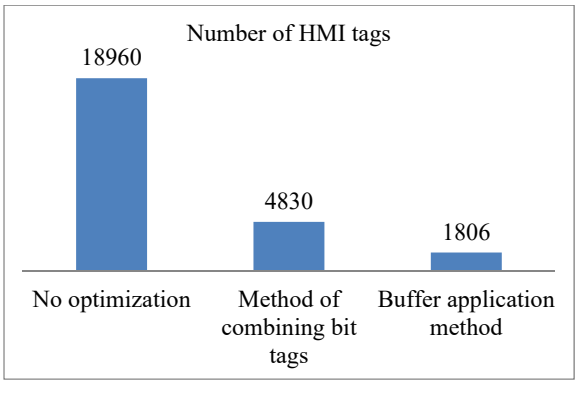

Fig. 4. Comparison of the number of tags before optimization and after using the technique for combining bit variables, after using the buffer

The optimization coefficient for this technique is calculated as follows:

$$K_{opt} = \frac{18,960}{1,806} = 10.5. \quad (3)$$

Despite the significant saving of resources, the use of the buffer is accompanied by a number of limitations. The most significant limitation is the impossibility of using a buffer of 2 or more HMI tools. When used simultaneously, the buffer is "selected" by the last user. In the ISA-88 standard, this use is termed allocation of a resource with exclusive access, however, in practice, to implement the impossibility of selection, quite complex algorithms must be developed, so the buffer is "selected" by those who requested it without any blocking.

Another disadvantage of exchanging configuration data through a buffer is the rejection of tabular views of PLC maps and technological variables. In practice, there are solutions that allow one to bypass this limitation, but this option requires significant costs for writing scripts from the SCADA/HMI side, which is not always possible.

Another problem with using a buffer is that there are often elements on the human-machine interface that visualize certain configuration data. And the use of a buffer in this case makes such visualization impossible.

For example, an element of visualization of an analog variable is shown in Fig. 5.

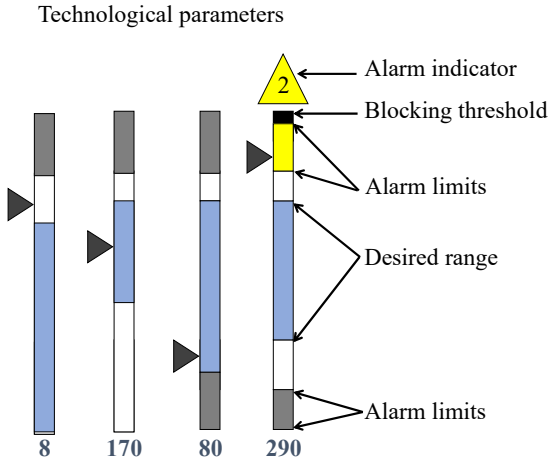

Fig. 5. An example of visualization of the display of an analog variable

Such indicators are highly effective from the point of view of transmitting information about the state of the parameter because in addition to the actual value of the parameter, the user is presented with additional information in the form of limits of normal functioning. The limits of triggering accidents of the HiHi, Hi, Lo, LoLo type are given in [17–21]. The problem is that such additional data in the proposed variant of optimization of the number of HMI tags can be obtained only through the buffer. Thus, only one such element can be displayed on the screen at a time.

Several solutions can be proposed for this problem.

Given that such parametric data rarely changes, it can be stored in internal SCADA/HMI tags that do not affect the exchange. In addition, the number of "external" HMI tags often appears in the resource limits – that is, those tags for which the data source is an external device. At the same time, this implementation requires the user to update them when, for example, new alarm limits are set via the configuration buffer. The disadvantage of this implementation is that often additional parameters that need to be displayed to the user can be changed not by the user, through the configuration buffer, but programmatically, according to the algorithm embedded in PLC. Accordingly, the proposed idea makes it impossible to update the data stored in the internal variables of the SCADA program.

The solution to the update problem can be implemented using a "shadow buffer". The task of the "shadow buffer" is to periodically update all the configuration data stored in the internal variables of the SCADA program without the user's participation. When implementing a "shadow buffer", another data structure responsible for the buffer is introduced into the system. The task of the HMI developer is to implement the cyclic loading of all variables into this buffer one by one, and the rewriting of data from the buffer into the corresponding set of internal SCADA variables. From a SCADA perspective, this can be implemented using a built-in scripting subsystem. As well as systems of planned tasks. This implementation solves the problem of updating data in internal variables. But its main drawback is the time it takes to update all variables. An experiment using 92 variables of the AIVAR type showed that it takes about 5 minutes to update all variables.

Another option for transferring configuration data, which must be periodically updated, can be implemented by packing this data into string variables of type string.

These approaches are successfully used by a number of integrators and are part of the PACFramework [21]. The use of the buffer requires additional improvement, which we are currently working on.

5. 3. 3. Using the configuration buffer to manage actuators

The essence of this technique is that in order to reduce the number of used HMI tags, it is possible to implement control over all actuators using a buffer. This will make it possible to opt out of the CMD variable in each data structure responsible for the execution mechanism. All control will be executed through the buffer variable. This technique has a significant drawback – with this implementation, only one actuator can be managed simultaneously with HMI. In the case of using an operator panel, this may still be acceptable. But in the case of full-fledged SCADA, when very often the operator needs to simultaneously open several pop-up windows to control various actuators, this approach is not acceptable.

The effect of using this technique is illustrated in Table 15.

Table 15

The result of optimization by using the configuration buffer to manage actuators

Type of data structure	Number of variables in the structure	Number of structures	Total number of HMI tags after optimization
DIVAR	1	700	700
DOVAR	1	400	400
AIVAR	3	100	300
AOVAR	3	20	60
VLVD	1	50	50
VLVA	3	20	60
DRV	3	30	90
DIVAR_BUFER	2	1	2
DOVAR_BUFER	2	1	2
AIVAR_BUFER	21	1	21
AOVAR_BUFER	9	1	9
VLVD_BUFER	2	1	2
VLVA_BUFER	5	1	5
DRV_BUFER	5	1	5
Total			1706

A comparative diagram of the number of tags before and after optimization in this way is shown in Fig. 6.

The optimization factor for this technique is:

$$K_{opt} = \frac{18,960}{1,706} = 11.11. \quad (4)$$

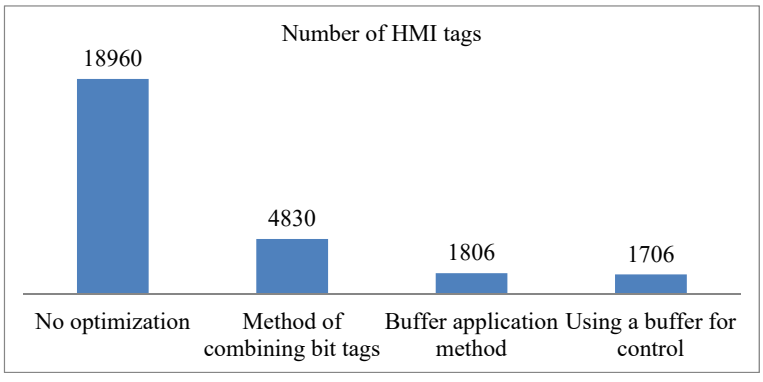

Fig. 6. Comparison of the number of HMI tags when using a buffer to control actuators

5. 3. 4. Using string variables of type STRING to optimize the number of used tags

Many PLCs and SCADA/HMIs have the ability to convert numbers to strings and vice versa. This makes it possible to store numbers as strings and convert them back to numbers when needed. The peculiarity of the string data type from the point of view of the SCADA program is that very often one HMI tag of the string data type allows storing up to 255 characters. On the one hand, it allows for serialization, similar to CSV or JSON in the IT world, and it also reduces the number of I/O tags.

The procedure is as follows: several numbers to be stored must be concatenated into a string, separated by, for example, a semicolon or any other separator. For the reverse procedure, it is necessary to break this string using a delimiter and get back the original numbers.

While this approach may prove useful in some scenarios, it also has limitations and drawbacks, which further add to the limitations of serialization:

– complexity of working with data: Working with numbers in the form of strings can be more difficult and less efficient than working with them in their original numerical format;

– increased probability of errors: There is an increased probability of errors when processing strings, especially if the format of the string is not followed or it contains incorrect data.

In general, the algorithm for using this technique is as follows:

– on the part of PLC, the conversion of numerical data into string data types;

– concatenation of several string variables into one, a separator is inserted between the variables (for example, a semicolon);

– transfer of a variable to SCADA/HMI in the form of an HMI tag;

– from the side of the SCADA program, with the help of the script subsystem, the HMI tag of the string type is separated, and the symbols are converted into output numerical data;

– received numerical values are stored in internal SCADA tags.

It is worth noting that in practice, the use of this algorithm contains a number of features that determine the limits of the use of this approach:

– the technique has the feasibility of use only in the case when the internal variables of the SCADA program are not limited in number, for example, by the cost of the license;

– the technique is convenient only for reading variable values from PLC, but not for writing new values;

– it is necessary to ensure a stable cycle of execution of this algorithm on the part of the SCADA program. And thereby provide the system with up-to-date data;

– there is a feature of saving floating point numbers. The peculiarity is that in the original form such variables can occupy a large number of characters. Therefore, to concatenate several floating-point numbers, it is necessary to know the predicted size of these variables. For the universality of the technique, one can specify a certain redundant format for saving floating-point numbers, which will be suitable for saving all variables and have a certain number of characters, and all floating-point variables should be rounded to this format.

In this study, it is proposed to choose the results of optimization of the number of HMI tags for the technique "Using the configuration buffer to control actuators" as the initial data. In this way, the control over actuators, the configuration of variables is proposed to be performed through a buffer, without applying the technique of packing into a string variable. And the rest of the variables, which are only needed for display and do not require changes by SCADA, are implemented using this technique.

Separately, worth specifying is the format for saving floating-point variables. For universality, let's set one format for all variables, which would ensure data transmission with the required accuracy. The format is chosen as follows:

$$S\#\#\#\#,\#\#\#\,, \tag{5}$$

where "*S*" defines a+or – sign, and the # sign indicates the number of characters.

If necessary, one can change the format individually for some variable separately.

The following information was used to determine the number of string variables to transfer all necessary variables:

– up to 255 characters can be transferred in one string variable;

– one UDINT type variable occupies 9 characters. When transmitting in the hexadecimal number system, the maximum value is FFFFFFFF (8 characters) + 1 separator ";";

– one variable of type REAL takes 11 characters – 10 characters from the formula 5+1 separator ";".

Thus, the number of used HMI tags of the STRING type is given in Table 16.

Taking into account that buffer variables were not converted, the total number of HMI tags will be equal to that given in Table 17.

A comparative diagram of the number of tags before and after optimization in this way is shown in Fig. 7.

The optimization factor for this technique is:

$$K_{opt}=\frac{18,960}{112}=169.286. \tag{6}$$

Thus, this technique showed an extremely high degree of optimization of the number of HMI tags. But it is worth noting that this technique already included previous optimization techniques. It should also be noted that this technique is the most resource-intensive in terms of both SCADA development and PLC application development.

Table 16

The number of used HMI tags of type STRING

Type of data structure	Number of variables in the structure	Type of data	Number of data structures	Number of variables to be packed into a string type	Number of characters in 1 variable	Total characters	The required number of variables of type STRING
DIVAR	1	UDINT	700	700	9	6,300	25
DOVAR	1	UDINT	400	400	9	3,600	15
AIVAR	1	UDINT	100	100	9	900	4
	2	Real		200	11	2,200	9
AOVAR	1	UDINT	20	20	9	180	1
	2	Real		40	11	440	2
VLVD	1	UDINT	50	50	9	450	2
VLVA	1	UDINT	20	20	9	180	1
	2	Real		40	11	440	2
DRV	1	UDINT	30	30	9	270	2
	2	Real		60	11	660	3
Total							66

Table 17

The result of optimization using string variables of type STRING

Type of data structure	Number of variables in the structure	Number of structures	Total number of HMI tags after optimization
DIVAR	1	700	25
DOVAR	1	400	15
AIVAR	2	100	13
AOVAR	2	20	3
VLVD	1	50	2
VLVA	2	20	3
DRV	2	30	5
DIVAR_BUFER	2	1	2
DOVAR_BUFER	2	1	2
AIVAR_BUFER	21	1	21
AOVAR_BUFER	9	1	9
VLVD_BUFER	2	1	2
VLVA_BUFER	5	1	5
DRV_BUFER	5	1	5
Total	–	–	112

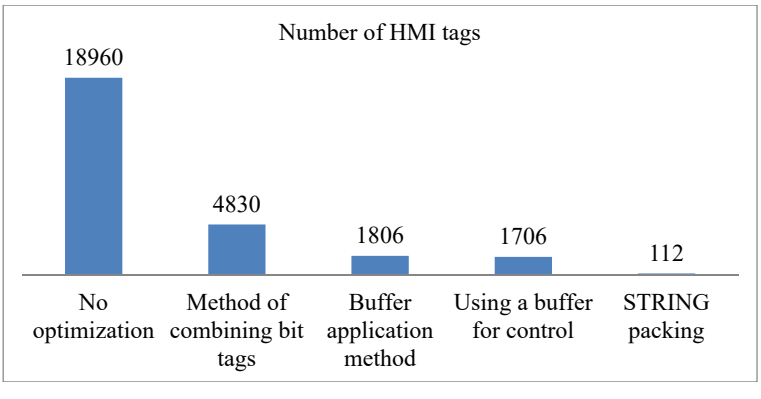

Fig. 7. Comparing the number of tags when using the string data type packing technique

The advancement of this technique is the implementation of configuration data exchange using approaches similar in nature to HTTP REST.

6. Discussion of results of investigating the techniques to optimize the number of tags

The technique of combining bit tags (bool) into double unsigned variables showed the result of reducing the number of HMI tags from 18,960 to 4,830 (Table 11) (by 4 times) (Fig. 2).

The value of the coefficient will fundamentally differ depending on the number of bool type tags. For 100 % of such tags only, the coefficient increased to 16 or 32, depending on the size of the variable packaging. Therefore, this approach has a very high efficiency when using a large number of tags of the bool type.

The advantages of this technique include relative simplicity, versatility, i.e., the ability to use it for various SCA-

DA programs/operator panels. A side positive effect is the consistency of bit states since they are always transmitted by the same tag, which the exchange of bit variables may not guarantee.

The technique of using the configuration buffer showed the result of reducing the number of HMI tags from 18,960 to 1,806 (10 times) (Table 14, Fig. 4).

This reduction is due to the use of one set of variables of the same type to multiplex many parametric variables that must be used only on demand.

This technique includes technique 1 and supplements it with the presence of a buffer (Fig. 3).

The technique of using a configuration buffer to control actuators (Table 15) is an evolution of technique 2, the essence of which is that the buffer is used not only for configuring objects but also to control actuators (Fig. 6). Unlike technique 2, real-time data is also multiplexed here, and therefore the coefficient increases depending on the number of multiplexed objects and can reach tens of times.

The technique of using string variables of type STRING showed the result of reducing the number of HMI tags from 18,960 to 112 (more than 150 times) (Table 16). This is explained by the use of commonly accepted IT approaches termed serialization and archiving.

A separate advantage of the technique for using string variables, in addition to extremely high optimization efficiency (Fig. 6), is that it can be used to solve the problem of displaying configuration parameters on visualization screens without using a shadow buffer.

The task of reducing the number of HMI tags is important in two aspects:

– reducing the load on communication channels and, accordingly, increasing the bandwidth;

– reducing the number of HMI tags when building a human-machine interface for operator panels, where the maximum number of HMI tags is strictly limited;

– reducing the number of HMI tags when implementing SCADA systems, where the number of tags is limited by the existing license of the execution environment.

As noted above, a high-performance human-machine interface requires a large amount of context data and configuration parameters that circulate between SCADA/HMI and PLC, which in turn leads to the need for a large number of human-machine interface resources, which are not always available. Reducing the number of tags/input-output solves this problem, as it is one of the parameters that directly affects the resources used. Our study provides several ways to do this.

The disadvantages and limitations of these optimization techniques include the following.

In the technique of combining bit tags (bool) into double unsigned variables, the following restrictions can be attributed: the situation when the SCADA program or the operator panel does not allow access to specific bits in the HMI thesis. Also, the situation when SCADA/HMI when working with alarms does not make it possible to use a separate bit in the tag as a trigger.

The technique to use a configuration buffer, compared to the technique to combine bit tags (bool), into double unsigned variables, has more significant disadvantages. In particular, the impossibility of using a buffer of 2 or more HMI tools at the same time. This shortcoming can be compensated by increasing the number of buffers from one to the required number of HMI devices or make an asynchronous buffer (without allocation). Limitations include the display of configuration parameters on visualization screens, for example, the inability to display value ranges for analog variables. This limitation can be solved by the shadow buffer technique.

In the technique of using the configuration buffer to control the actuators, compared to the technique of using the configuration buffer, the feature is that it is not possible to control several actuators at the same time – this is a significant disadvantage. It is not recommended for use without significant reasons, as it significantly limits the use of HMI.

In the technique of using string variables of the STRING type, the main limitations are the demand for computing power, as well as the presence of a developed scripting subsystem. Therefore, this technique is appropriate for use only for advanced SCADA programs and has a limited possibility of use in operator panels. The disadvantage of this approach is the use of significant computing resources, which negatively affects the speed of processing and the complexity of development and maintenance.

These techniques demonstrate high efficiency of optimization but require more time for implementation from the developer. It is also worth noting that our paper does not provide an exhaustive list of techniques but only those that we have used in practice. In addition, work is currently underway to improve some of the techniques.

The effectiveness of the above techniques was evaluated exclusively in terms of the number of input/output tags, which is one indicator of the used resources, but not the only one.

In the future, it is worth analyzing the above techniques in relation to their impact on other properties of the automated control/management system. In particular, such indicators include the impact on communication exchange, the impact on direct indicators of work productivity, and the impact on system reliability. Empirically, the communication load as the number of tags is reduced should decrease as the number of tags is reduced. This in turn can increase the bandwidth of the communication channel and therefore potentially reduce the response time of the system when needed. However, some of the above techniques require additional computing resources, which may have the opposite effect. This requires experimental studies for several typical configurations. The most challenging task here is the selection of relevant typical sets of network protocols, PLC, and SCADA/HMI facilities in various configurations. All these component systems have a significant impact on the results of experiments, so they can differ tens and hundreds of times depending on the chosen configuration and adopted development approaches. Even the version or subversion of the software or hardware significantly affects the performance. The same can be said about the reliability of the system. Direct indicators of work productivity mean criteria that characterize the final functionality of the system, in particular the speed of updating data on the display mnemonics, the speed of the system's response to user action, page update time, etc. All the difficulties of network exchange analysis are typical for this study; however, an additional challenge is also to define techniques for measuring indicators for further criteria calculations.

The above list of techniques is not exhaustive, it is worth thinking about other, alternative approaches or their combination.

7. Conclusions

1. We have systematized and classified HMI I/O tags according to their purpose. That has made it possible to highlight the necessary variables used in the system, to determine how and when they are involved in the communication between the PLC and SCADA/HMI, and how they could be optimized to reduce the total number.

In particular, it was determined that:

– control data requires a lot of additional contextual data that complements the main measurement data with status information;

– contextual data should be kept in single structures with controlled and managed variables, they are updated in real time;

– for the grouping of main variables and context variables, worth using are the approaches from modern standards based on Equipment;

– additional data are configuration parameters: their number is many times greater than real-time data; they need to be updated on demand, not in real time, so they should be separated in the exchange.

2. Typical data structures used in PLCs to control various types of objects were selected for the study. Among the typical structures that cover most of the typical tasks, the following were identified as the most used:

– responsible for discrete measuring (input for PLC) and controlled (output for PLC) parameters;

– responsible for analog measuring (input for PLC) and controlled parameters (output for PLC);

– responsible for actuators: shut-off and regulating valves, engine drives.

A list of typical status (contextual) and configuration parameters for structures has been compiled, in particular:

– measurement/control data;

– position data;

– alarm status bits;

– mode bits;

– commands;

– processing function setting parameters;

– setting parameters of actuators;

– alarm setting parameters;

– service commands.

3. The effectiveness of techniques that could make it possible to reduce the number of tags in the conditional automation object has been studied. The optimization coefficient was calculated for each technique:

– for the technique of combining bit tags in (bool), in double unsigned variables (UDINT/ULONG/DWORD), the optimization factor is 3.93;

– for the technique using the configuration buffer – 10.5;

– for the technique using the configuration buffer to control actuators – 11,11;

– for the technique of using string variables of type STRING to optimize the number of used HMI tags – 169,286.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Pupena, O. M. (2020). Rozroblennia liudyno-mashynnykh interfeisiv ta system zbyrannia danykh z vykorystanniam prohramnykh zasobiv SCADA/HMI. Kyiv: LIRA-K, 594. Available at: https://lira-k.com.ua/preview/12760.pdf
2. Shyshak, A., Pupena, O. (2020). Management of human-machine interface lifecycle. Scientific Works of National University of Food Technologies, 26 (3), 17–27. https://doi.org/10.24263/2225-2924-2020-26-3-4
3. Urbas, L., Obst, M., Stöss, M. (2012). Formal Models for High Performance HMI Engineering. IFAC Proceedings Volumes, 45 (2), 854–859. https://doi.org/10.3182/20120215-3-at-3016.00151
4. Panter, L., Leder, R., Keiser, D., Freitag, M. (2024). Requirements for Human-Machine-Interaction Applications in Production and Logistics within Industry 5.0 – A Case Study Approach. Procedia Computer Science, 232, 1164–1171. https://doi.org/10.1016/j.procs.2024.01.114
5. Crompton, J. (2021). Data management from the DCS to the historian and HMI. Machine Learning and Data Science in the Power Generation Industry, 93–122. https://doi.org/10.1016/b978-0-12-819742-4.00005-6
6. Šverko, M., Grbac, T. G. (2024). Automated HMI design as a custom feature in industrial SCADA systems. Procedia Computer Science, 232, 1789–1798. https://doi.org/10.1016/j.procs.2024.02.001
7. Scaife, R. (2016). Control system interface design. Human Factors in the Chemical and Process Industries, 223–239. https://doi.org/10.1016/b978-0-12-803806-2.00013-3
8. Mustafa, F. E., Ahmed, I., Basit, A., Alvi, U.-E.-H., Malik, S. H., Mahmood, A., Ali, P. R. (2023). A review on effective alarm management systems for industrial process control: Barriers and opportunities. International Journal of Critical Infrastructure Protection, 41, 100599. https://doi.org/10.1016/j.ijcip.2023.100599

9. Rao, H. R. M., Zhou, B., Brown, K., Chen, T., Shah, S. L. (2024). Alarm correlation analysis with applications to industrial alarm management. Control Engineering Practice, 143, 105812. https://doi.org/10.1016/j.conengprac.2023.105812
10. Cruz-Benito, J., García-Peñalvo, F. J., Therón, R. (2019). Analyzing the software architectures supporting HCI/HMI processes through a systematic review of the literature. Telematics and Informatics, 38, 118–132. https://doi.org/10.1016/j.tele.2018.09.006
11. Zhou, C., Su, H., Tang, X., Cao, Y., Yang, S. (2024). Global self-optimizing control of batch processes. Journal of Process Control, 135, 103163. https://doi.org/10.1016/j.jprocont.2024.103163
12. Pupena, O., Mirkevych, R., Klymenko, O. (2020). Praktychni rekomendatsiyi do realizatsiyi elementiv standartu IEC 61512 v prohramnomu zabezpechenni system keruvannia. TEKhNIChNYI KOMITET 185 «PROMYSLOVA AVTOMATYZATsIIa». Kyiv. Available at: https://tk185.appau.org.ua/guide/aCampus-users-guides-IEC61512+++.pdf
13. Pupena, O., Klymenko, O., Mirkevych, R. (2020). Pryntsypy funktsionuvannia system keruvannia osnovnym vyrobnytstvom cherez pryzmu standartu IEC-62264. TEKhNIChNYI KOMITET 185 «PROMYSLOVA AVTOMATYZATsIIa». Kyiv. Available at: https://tk185.appau.org.ua/guide/aCampus-users-guides-IEC62264+++.pdf
14. Rockwell Automation Process HMI Style Guide. White Paper. Rockwell Automation. Available at: https://literature.rockwellautomation.com/idc/groups/literature/documents/wp/proces-wp023_-en-p.pdf
15. Best Design Practices: How to Create High Performance HMI to Enhance Operator Efficiency (2023). White Paper. Movicon. NExT. Available at: https://www.tug.at/images/news/whitepapers/Best_practices_for_high_performance_HMI_design_White_Paper_ENUS_2023-03-24.pdf
16. PAC Framework V1. Available at: https://github.com/pupenasan/PACFramework
17. Bhole, M., Kastner, W., Sauter, T. (2023). Knowledge Representation of Asset Information and Performance in OT Environments. 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA). https://doi.org/10.1109/etfa54631.2023.10275721
18. Peco Chacón, A. M., García Márquez, F. P. (2019). False Alarms Management by Data Science. Data Science and Digital Business, 301–316. https://doi.org/10.1007/978-3-319-95651-0_15
19. Peco Chacón, A. M., García Márquez, F. P. (2021). False Alarm Detection in Wind Turbine Management by Tree Model. Lecture Notes on Data Engineering and Communications Technologies, 543–553. https://doi.org/10.1007/978-3-030-79203-9_42
20. Qiu, Y., Feng, Y., Infield, D. (2020). Fault diagnosis of wind turbine with SCADA alarms based multidimensional information processing method. Renewable Energy, 145, 1923–1931. https://doi.org/10.1016/j.renene.2019.07.110
21. Asaadi, M., Izadi, I., Hassanzadeh, A., Yang, F. (2022). Assessment of alarm systems for mixture processes and intermittent faults. Journal of Process Control, 114, 120–130. https://doi.org/10.1016/j.jprocont.2022.04.002