OULUN AMMATTIKORKEAKOULU

Mika Karhumaa

TWINCAT 3 HMI SUITABILITY FOR AUTOMATION SOLUTIONS

TWINCAT 3 HMI SUITABILITY FOR AUTOMATION SOLUTIONS

Mika Karhumaa
Final projects
Spring 2023
Electrical and Automation Engineering
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Electrical and Automation Engineering, Automation

Author: Mika Karhumaa
Title of the thesis: TwinCAT 3 HMI suitability for automations solutions
Thesis examiner(s): Manne Tervaskanto, Marko Pöyskö
Term and year of thesis completion: Spring 2023 Pages: 40 + 10 appendices

The goal of this thesis was to find if TwinCAT 3 HMI is capable for JOT's automation projects. The purpose of the thesis was to make a human machine interface and study its features. The software and methods used are introduced in this thesis.

JOT wanted to have lighter operating environment for the HMI. The new HMI could bring some new features and make the work of software engineers easier. My task was also to get know the features of the TE2000 package, because it has not been used before by JOT. The source of information regarding the user interface package was the Beckhoff information system.

A user friendly and highly responsive user interface was made in the thesis. User controls, JavaS-cript functions, and the controls from the package were used. The features of the user interface included a functional alarm list, a user management system, localization, data visualization and an Input/Output list. The alarms system and the data visualization can be developed in the future. It is possible to add and alarm error bypass and retry features. The data is also possible to be retrieved from the SQL-server.

The suitability of the TwinCAT 3 HMI to the JOT's projects was established. The development environment itself if free to use, but the published HMI needs a licence. The price of the licences needs to be noticed, when TwinCAT 3 HMI is used in a project.

Keywords: TwinCAT 3, HMI, Human machine interface, TE2000

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Sähkö- ja automaatiotekniikan tutkinto-ohjelma, automaatiotekniikka

Tekijä: Mika Karhumaa
Opinnäytetyön nimi: TwinCAT 3 HMI:n soveltuvuus automaatioratkaisuihin
Työn ohjaajat: Manne Tervaskanto, Marko Pöyskö
Työn valmistumislukukausi ja -vuosi: Kevät 2023 Sivumäärä: 40 + 10 liitettä

Tämän opinnäytetyön tavoitteena oli selvittää TwinCAT 3 HMI:n soveltuvuus tilaajan automaatioprojekteihin. Työn tilaaja oli JOT Automation Oy. Opinnäytetyön tarkoituksena oli tehdä käyttöliittymä ja tutkia sen ominaisuuksia.

Keskeisin tavoite oli saada JOT:in nykyistä käyttöliittymää kevyempi toimintaympäristö. Uusi käyttöliittymä voisi tuoda uusia ominaisuuksia ja helpottaa ohjelmistoinsinöörien työtä. Työssä tuli myös tutustua TE2000-paketin ominaisuuksiin, koska kyseinen ohjelmisto ei ole vielä ollut JOT:illa käytössä. Pääasiallinen tiedonlähde käyttöliittymäpakettiin liittyen oli Beckhoff Information System.

Opinnäytetyössä saatiin tehtyä käyttäjäystävällinen ja nopeasti reagoiva käyttöliittymä. Työssä hyödynnettiin TwinCAT 3 TE2000 HMI paketin käyttäjäohjausobjekteja, JavaScript funktioita ja paketissa valmiina olevia ohjaimia. Käyttöliittymän ominaisuuksiin saatiin toimiva hälytyslista, käyttäjähallintajärjestelmä, lokalisointi, datan visualisointi sekä Input/Output-lista. Mahdollisia kehityskotia työssä on hälytysjärjestelmässä ja datan visualisoinnissa. Hälytysjärjestelmään on mahdollista tehdä virheen ohitus- sekä uudelleenyritysominaisuus. Lisäksi datan hakeminen SQL-serveriltä on mahdollista.

Työn avulla pystyttiin toteamaan TwinCAT 3 HMI:n soveltuvuus tilaajan projekteissa. Kehitysympäristö on ilmainen, mutta julkaistu käyttöliittymä tarvitsee lisenssin. Projekteissa lisenssien hinta tulee ottaa huomioon.

Asiasanat: TwinCAT 3, HMI, käyttöliittymä, TE2000

CONTENTS

APPENDICES

LIST OF ABBREVIATIONS

ADS: Automation Device Specification

CSS: Cascading Style Sheets

HMI: Human Machine Interface

HTML: HyperText Markup Language

IEC: International Electrotechnical Commission

IO: Input/Output

OPC UA: Open Platform Communications Unified Architecture

PC: Personal computer

PLC: Programmable Logic Controller

SQL: Structured Query Language

XAE: eXtended Automation Engineering

1 INTRODUCTION

Efficient and simple software are important for software engineers. Keeping used software in minimum decreases repeating work and makes working more efficient.

The subject of the thesis was to make HMI equivalent to the current one with TwinCAT HMI package. JOT's current HMI is made with Windows presentation foundation or WPF. WPF is heavy and requires its own different program, when using it with TwinCAT projects. The objective was to make both visually and functionally similar HMI. Most important features were working alarms system, displaying data from SQL-server, user management, and automatically populating input/output-list.

I had some experience with TwinCAT XAE, but I had not used HMI package before. The Beckhoff information system web page was used as source of information from software features and different options.

Basic TwinCAT HMIs can be made very easily with TwinCAT software. This thesis focuses on the difficult and more complicated parts of the HMI. The study about the features and licences were needed to make working solution.

2 JOT AUTOMATION LTD

JOT Automation Ltd is part of Victory Precision Manufacture Co Ltd. JOT was founded in 1988. JOT's headquarters are located in Oulu, but it also operates in the Asia, America, and Europe. Currently there are over 300 employees in 9 different countries. Name JOT comes from words just on time. It is used to describe production with non to minimal intermediate storage. (1).

JOT offers production solutions for assembly, testing, process, material handling, and custom automation. There are standard products for material handling, testing, assembly, and process. Those products always come with key features, but JOT offers many optional and custom features for customer needs. (2.)

3 BECKHOFF TWINCAT

3.1 TwinCAT 3 eXtended Automation Engineering (XAE)

TwinCAT 3 XAE is free development tool for PLC programming. TwinCAT has advantage over its competitor for having PC based solution integrated into Microsoft Visual Studio. Program supports many modular extensions and third-party components. TwinCAT functions can be used to create project-specific solution. Development environment supports C, C++, and MATLAB/Simulink programming in addition with all IEC 61131-3 languages. Different interfaces can connect TwinCAT to wide range of protocols and extend its use even more. TwinCAT XAE has basic visualization components for making HMI or testing projects. (3).

3.2 TwinCAT 3 HMI Engineering TE2000

TwinCAT 3 TE2000 is one of the many components of engineering package. It can be used to create HMI with most recent web technologies. HMI package supports HTML5, JavaScript and TypeScript languages. Those technologies give HMI to responsive and adaptive features. (4). HMI can be accessed with all devices that has HTML5-capable browser. Using standard controls, user can make and configure HMI with graphical editor. For more demanding functions or visually different controls, the user can make own user controls. (5.)

3.2.1 Human machine interface (HMI)

HMI is user interface that is used to connect PLC and user. The term is mostly used in industrial process, although it can be applied to any kind of screen. HMIs have a lot of features to help operators keep machine or plant in control. Depending on the hardware HMIs can visually display machine status, productivity or basically any information from PLC. Maintenance person can use HMI to diagnose fault or in advance maintenance tasks. (6.)

3.2.2 JavaScript

JavaScript is commonly used programming language. Web developers use it to create more dynamic interactions for servers, games, applications, or web pages. JavaScript is lightweight and it is used side by side with HTML and CSS. Benefits over its competitors are simplicity, speed, versatility, popularity, and continuous updates. (7.)

3.3 TwinCAT 3 HMI server

TwinCAT 3 TF2000 offers modular web server for TwinCAT HMI. Architecture is capable to be used with multiple clients, servers, or runtimes. TwinCAT ADS and OPC UA protocols enables communication with all TwinCAT devices. The HMI server supports custom extensions alongside with few ready server extensions. If the user needs more than one client for the server, those can be added with a client packs. (8.)

4 MAKING THE HMI

The project was meant to be copied visually and technically from JOT's current HMI, so planning did not take long time. The JOT's PLC-base for PLC project could also be used for this project. Main thing was to make HMI automatically adapt its elements to fit screen horizontally and vertically. Technical part was to make inputs/outputs list generate automatically so the project is easily reusable in a future project. The user management and localization were to be added. Also, is was needed to make alarms work with HMI and display data from SQL-server.

4.1 Main view

At start TwinCAT XAE has option to make different kinds of projects. Creating new HMI project gives some elements at start. Main view is named "Desktop.view" and it is opened by default in projects. Toolbox contains all the available controls that can be just dragged to view. Each element has its own properties that gives it wanted functions and looks.

The banners were made with rectangle from toolbox (Figure 1). As seen in Figure 2, the layout is 0 pixels from left, 100 from top and 20 from bottom. This keeps left banner always same distance from those points, and it looks same in both horizontal and vertical display. Width parameter also keeps it 200 pixels wide.

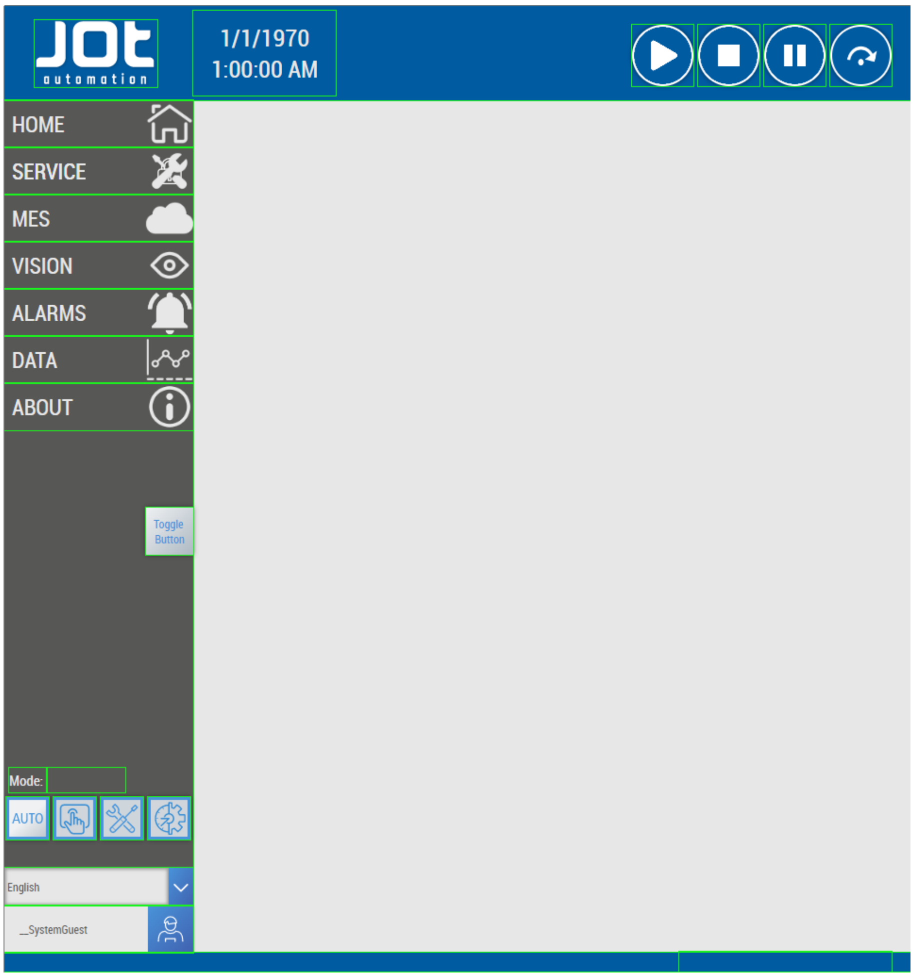
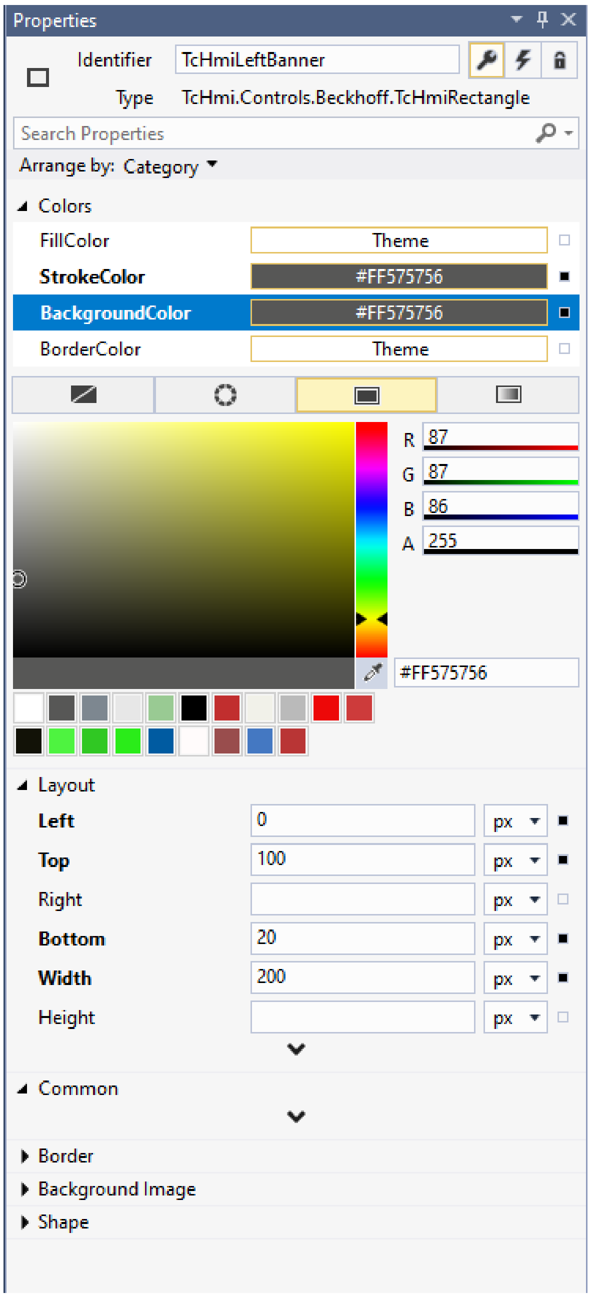


FIGURE 1. Desktop.view


Properties

Identifier TcHmiLeftBanner

Type TcHmi.Controls.Beckhoff.TcHmiRectangle

Search Properties

Arrange by: Category

Colors

FillColor	Theme
StrokeColor	#FF575756
BackgroundColor	#FF575756
BorderColor	Theme

R 87
G 87
B 86
A 255

#FF575756

Layout

Left	0	px
Top	100	px
Right		px
Bottom	20	px
Width	200	px
Height		px

Common

Border

Background Image

Shape

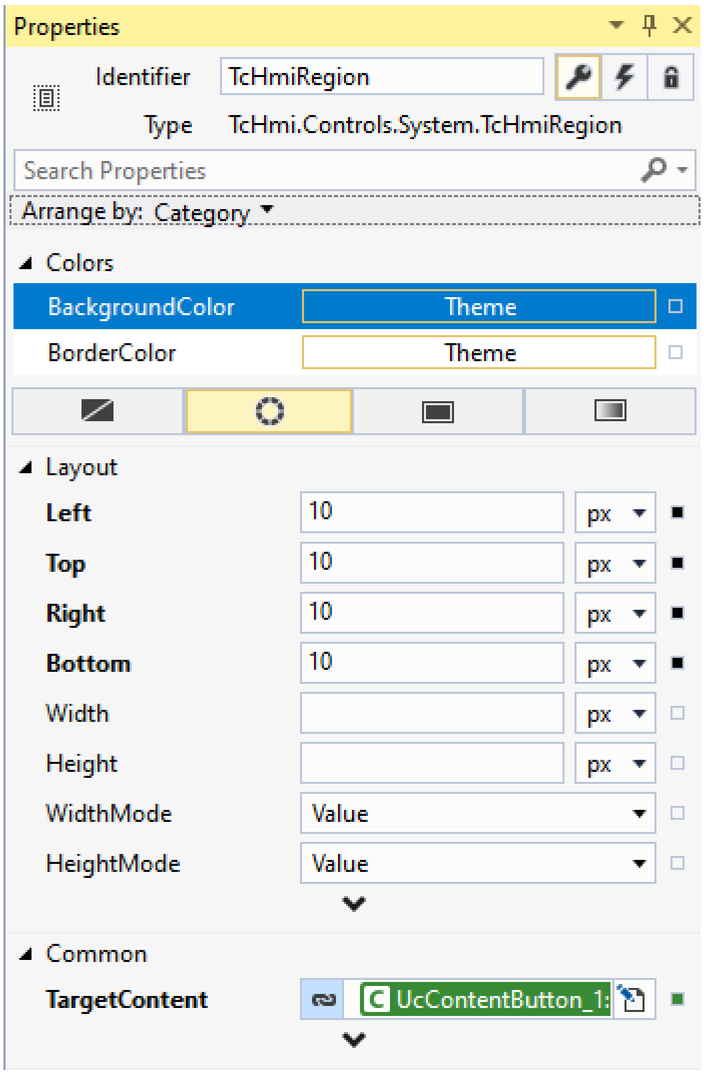
FIGURE 2. Left banner properties.

4.2 Navigation

Navigation keeps inside a few basic elements. Buttons are used to load content inside of the region. Multiple regions can be nested and make HMI wider.

4.2.1 Region

Regions can be used to display a content inside of it. To load new page, it is needed to give new page path to this "TargetContent" (Figure 3) property. (9.) In figure 3, the path for the Home-button content was given, so it automatically loads home page when initialized.



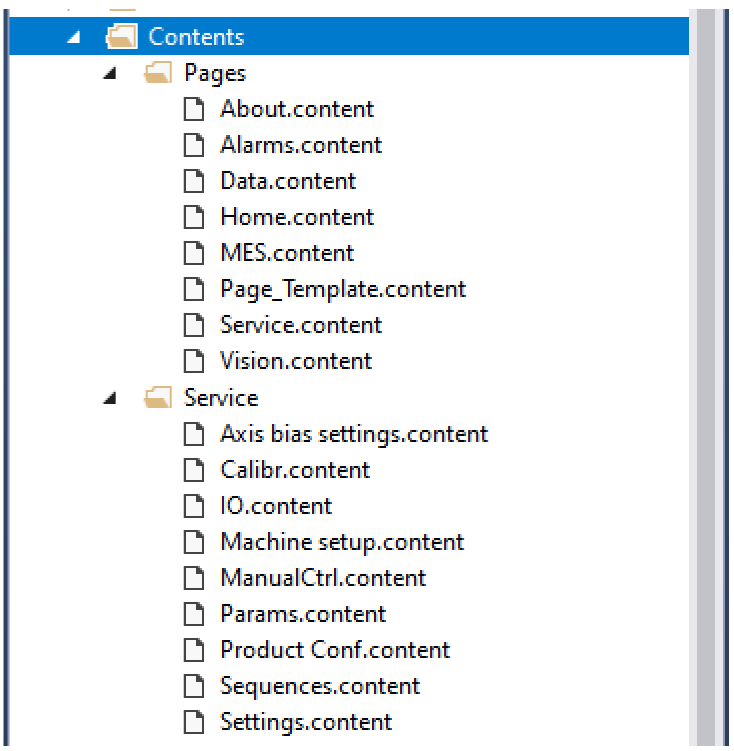
Identifier	TcHmiRegion	
Type	TcHmi.Controls.System.TcHmiRegion	
BackgroundColor	Theme	
BorderColor	Theme	
Left	10	px
Top	10	px
Right	10	px
Bottom	10	px
Width		px
Height		px
WidthMode	Value	
HeightMode	Value	
TargetContent	UcContentButton_1:	

FIGURE 3. Region properties

4.2.2 Content

A content is a container that can be loaded in a region. For example, multiple contents can subdivide operation into many different contents. (10.) This makes HMIs clearer for complex devices or solutions.

This project has 7 main pages and several subpages (Figure 4). Each page has its own elements. Those elements are used to display or control PLC variables and data.



- Contents
  - Pages
    - About.content
    - Alarms.content
    - Data.content
    - Home.content
    - MES.content
    - Page_Template.content
    - Service.content
    - Vision.content
  - Service
    - Axis bias settings.content
    - Calibr.content
    - IO.content
    - Machine setup.content
    - ManualCtrl.content
    - Params.content
    - Product Conf.content
    - Sequences.content
    - Settings.content

FIGURE 4. Contents

A new folder was added and named “Contents”. Right-click on the content folder and then select ADD -> New Item (Figure 5). Then it opens window (Figure 6) that can be used through whole project, when adding new items. Content is selected from the list and named “About.content”. This creates new content that can be edited for project needs.

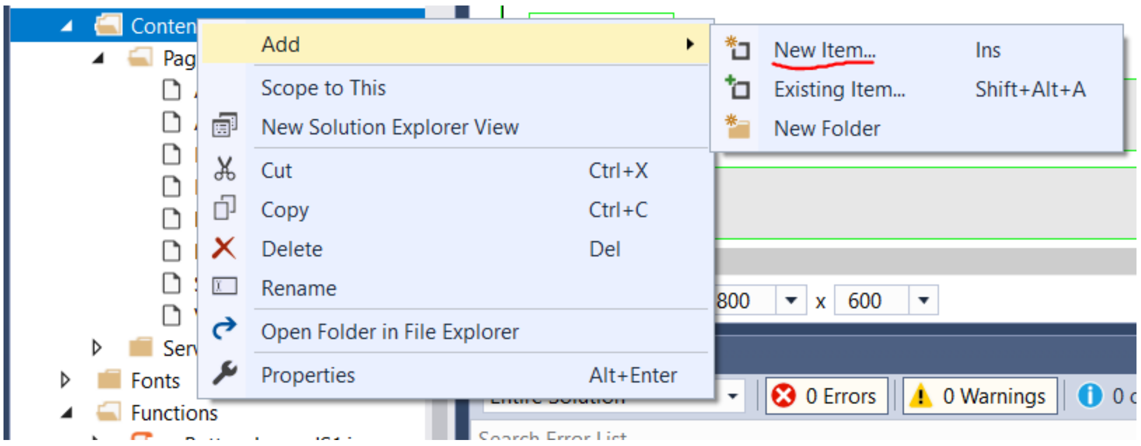
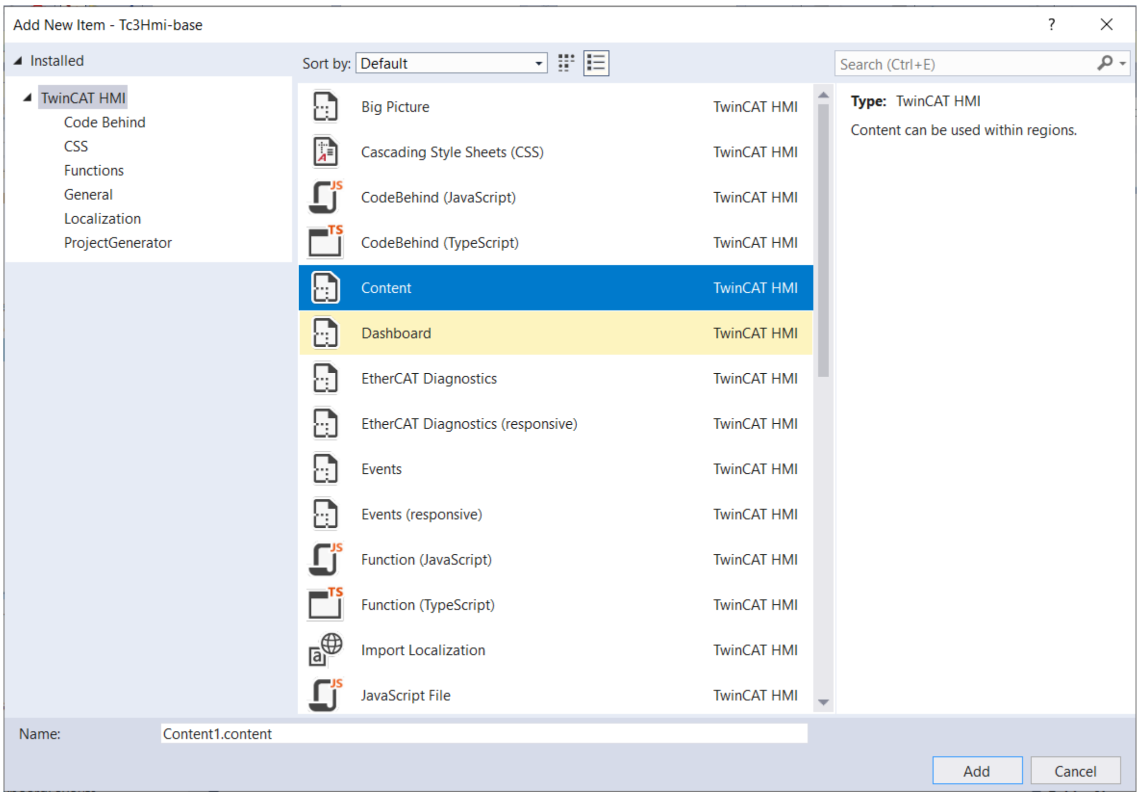


FIGURE 5. Add new item.


Add New Item - Tc3Hmi-base

Installed

Sort by: Default

Search (Ctrl+E)

TwinCAT HMI
Code Behind
CSS
Functions
General
Localization
ProjectGenerator

Big Picture	TwinCAT HMI
Cascading Style Sheets (CSS)	TwinCAT HMI
CodeBehind (JavaScript)	TwinCAT HMI
CodeBehind (TypeScript)	TwinCAT HMI
Content	TwinCAT HMI
Dashboard	TwinCAT HMI
EtherCAT Diagnostics	TwinCAT HMI
EtherCAT Diagnostics (responsive)	TwinCAT HMI
Events	TwinCAT HMI
Events (responsive)	TwinCAT HMI
Function (JavaScript)	TwinCAT HMI
Function (TypeScript)	TwinCAT HMI
Import Localization	TwinCAT HMI
JavaScript File	TwinCAT HMI

Type: TwinCAT HMI
Content can be used within regions.

Name: Content1.content

FIGURE 6. Add content.

The current JOT's HMI was used as a reference for all the pages. About page looks basically same in both platforms. Text blocks were added from toolbox. The text blocks parameters allowed to make them look similar as in the current HMI (Figure 7).

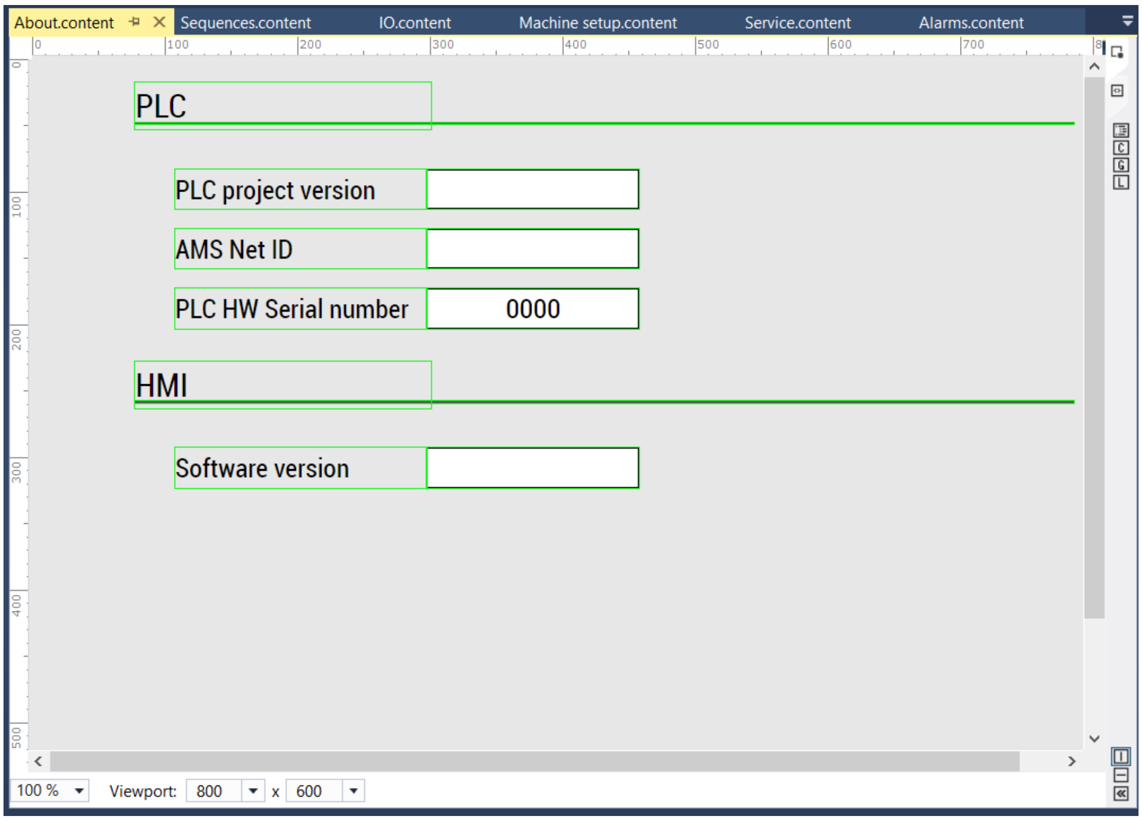


FIGURE 7. About.content.

The content of a text block can be changed in properties. In this case it was needed to create data binding for server variables. This enables text to automatically change, when for example PLC project version changes. Click on the square next to text opens menu to create data binding (Figure 8). The text was mapped to PLC1.GVL_Constant.Version (Figure 9).

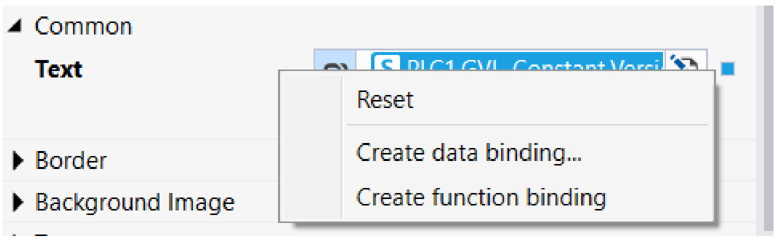
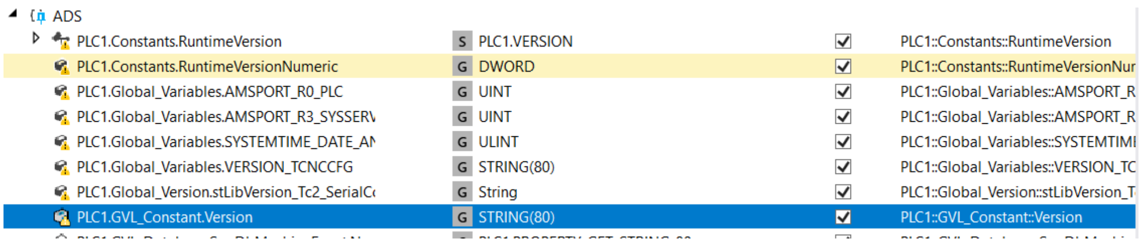


FIGURE 8. Create data binding.



ADS			
PLC1.Constants.RuntimeVersion	S PLC1.VERSION	✓	PLC1::Constants::RuntimeVersion
PLC1.Constants.RuntimeVersionNumeric	G DWORD	✓	PLC1::Constants::RuntimeVersionNur
PLC1.Global_Variables.AMSPORT_R0_PLC	G UINT	✓	PLC1::Global_Variables::AMSPORT_R
PLC1.Global_Variables.AMSPORT_R3_SYSSERV	G UINT	✓	PLC1::Global_Variables::AMSPORT_R
PLC1.Global_Variables.SYSTEMTIME_DATE_AN	G ULINT	✓	PLC1::Global_Variables::SYSTEMTIMI
PLC1.Global_Variables.VERSION_TCNCCFG	G STRING(80)	✓	PLC1::Global_Variables::VERSION_TC
PLC1.Global_Version.stLibVersion_Tc2_SerialC	G String	✓	PLC1::Global_Version::stLibVersion_T
PLC1.GVL_Constant.Version	G STRING(80)	✓	PLC1::GVL_Constant::Version

FIGURE 9. PLC project version.

4.2.3 User Control- buttons

Buttons from the toolbox have their own limits visually. The goal was to make buttons to look same as in current HMI. User controls enable option to make own buttons. These navigation buttons in Figure 10 are user controls.

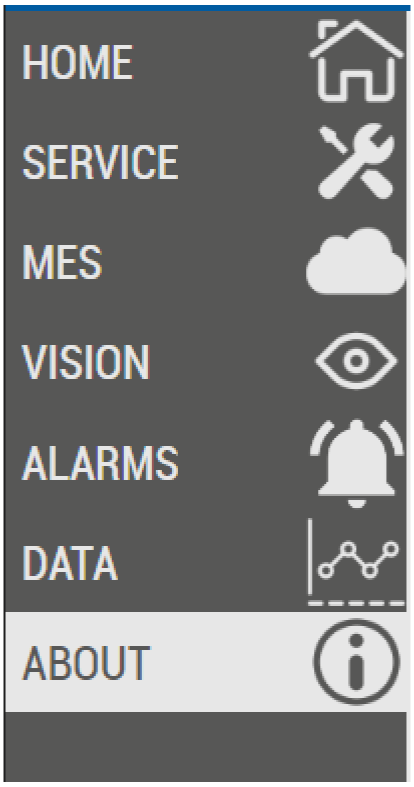
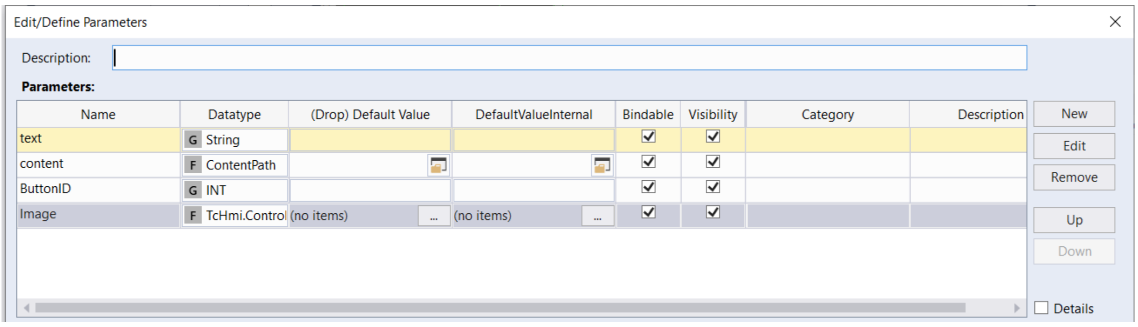


FIGURE 10. Navigation buttons.

A new folder was added and the user control inside of it. First step was to add variables to the user control (Figure 11). “Text” is string variable, that is used to link text of the text block to a property. “Content” variable is used to link contents path. Changing buttons background color, text color and image when pressed, needs JavaScript functions to work. For that use there is “ButtonID” variable. When there are multiple buttons, it is used to identify each one of them. “Image” is state list variable.

It is linked to a state image control. This control has state list option to give multiple images referring to a current state.



Edit/Define Parameters

Description:

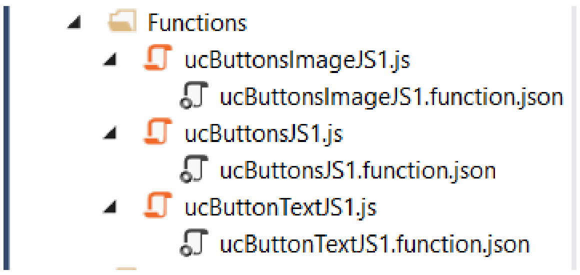
Parameters:

Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility	Category	Description
text	G String			✓	✓		
content	F ContentPath			✓	✓		
ButtonID	G INT			✓	✓		
Image	F TcHmi.Control	(no items) ...	(no items) ...	✓	✓		

New | Edit | Remove | Up | Down | ☐ Details

FIGURE 11. User control parameters

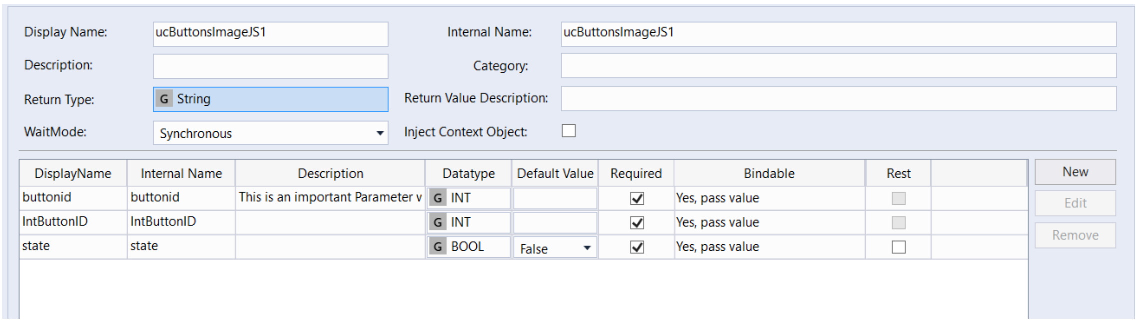
The goal was to make buttons match visually to current HMI's button. Buttons are 200 pixels wide and 50 pixels high. The text block and state image controls were added inside user control. Then the variables were linked to them. The colors of the button needed to change when it was toggled. And if one button is pressed, all other should not be toggled. This function was made in PLC, but it had delays. More responsive option was to make it with JavaScript. The JavaScript functions were added to the project (Figure 12). Each color (background, text, and image) needed their own function.



- Functions
  - ucButtonsImageJS1.js
    - ucButtonsImageJS1.function.json
  - ucButtonsJS1.js
    - ucButtonsJS1.function.json
  - ucButtonTextJS1.js
    - ucButtonTextJS1.function.json

FIGURE 12. Functions for color change

Each function needed three variables (Figure 13.). "IntButtonID" and "buttonid" is used to identify which button is currently pressed. "IntButtonID" is later mapped to an HMI project symbol described later. There is only 2 images in this case so the datatype Boolean can be used to change button's state.



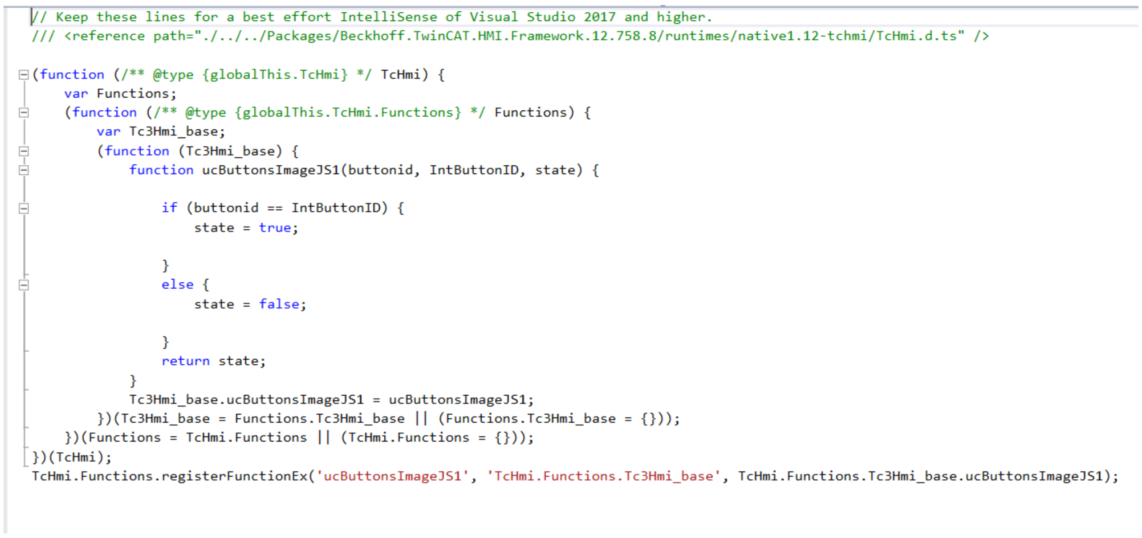
Display Name:	ucButtonsImageJS1	Internal Name:	ucButtonsImageJS1
Description:		Category:	
Return Type:	String	Return Value Description:	
WaitMode:	Synchronous	Inject Context Object:	☐

DisplayName	Internal Name	Description	Datatype	Default Value	Required	Bindable	Rest
buttonid	buttonid	This is an important Parameter v	INT		✓	Yes, pass value	☐
IntButtonID	IntButtonID		INT		✓	Yes, pass value	☐
state	state		BOOL	False	✓	Yes, pass value	☐

FIGURE 13. JavaScript variables

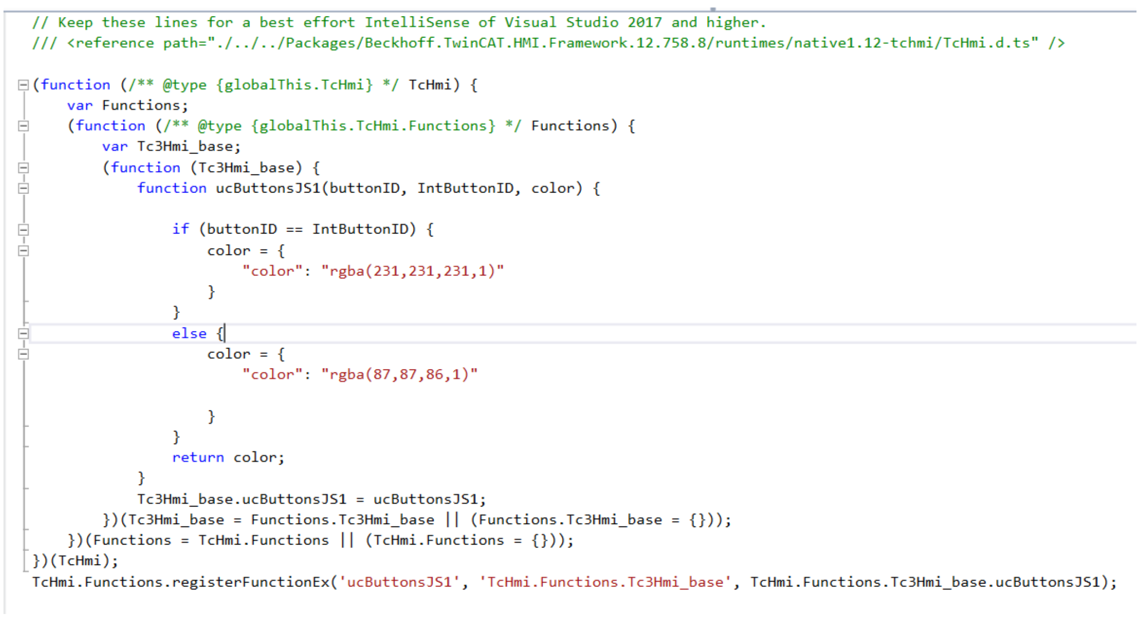
To make each button to change color and regions content when pressed, the internal symbol was added to the HMI project. This can be added from top banner TwinCAT HMI -> Windows -> Twin-CAT HMI Configuration. From there click on the "Internal Symbols" and "add my own symbol". Similar "ButtonID" variable was made as in Figure 13, but this time it can be used all over the HMI project. Datatype is also integer.

Figures 14, 15 and 16 show JavaScript code that changes buttons colors. In Figure 14 there is if condition. If "buttonid" is equal with "IntButtonID" then the statement is true, and "state" is set to true. Else "state" is set to false. Lastly script returns state value. Figures 15 and 16 are similar, but in those there is color codes. Those codes are retuned with similar statement as described before.



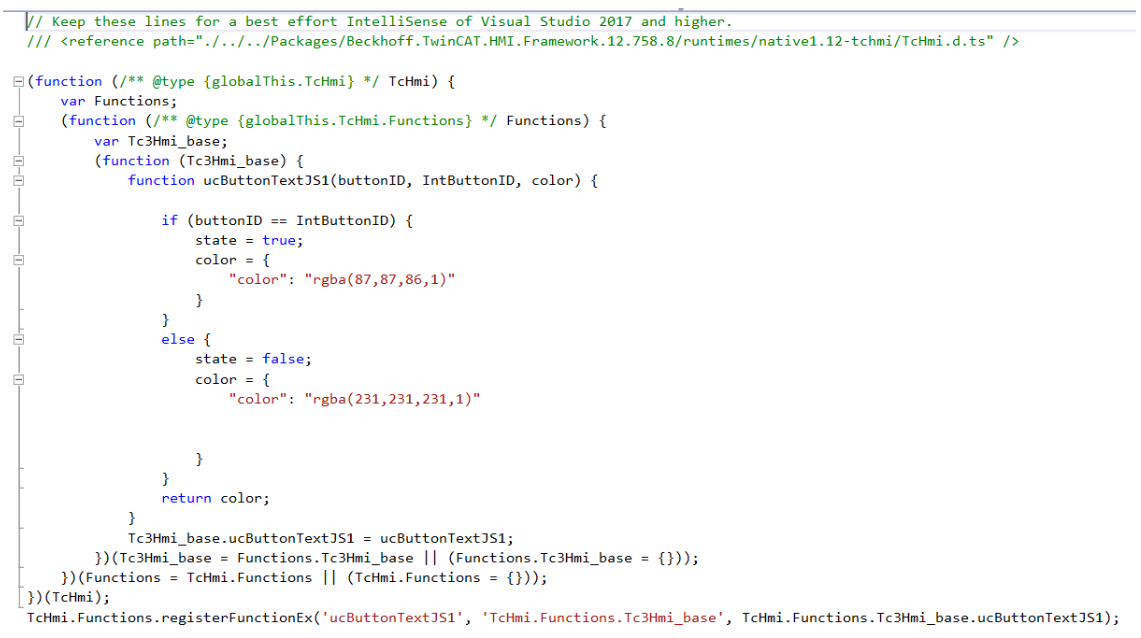
```
// Keep these lines for a best effort IntelliSense of Visual Studio 2017 and higher.
/// <reference path="./../../Packages/Beckhoff.TwinCAT.HMI.Framework.12.758.8/runtimes/native1.12-tchmi/TcHmi.d.ts" />

(function (/** @type {globalThis.TcHmi} */ TcHmi) {
    var Functions;
    (function (/** @type {globalThis.TcHmi.Functions} */ Functions) {
        var Tc3Hmi_base;
        (function (Tc3Hmi_base) {
            function ucButtonsImageJS1(buttonid, IntButtonID, state) {

                if (buttonid == IntButtonID) {
                    state = true;

                }
                else {
                    state = false;

                }
                return state;
            }
            Tc3Hmi_base.ucButtonsImageJS1 = ucButtonsImageJS1;
        })(Tc3Hmi_base = Functions.Tc3Hmi_base || (Functions.Tc3Hmi_base = {}));
    })(Functions = TcHmi.Functions || (TcHmi.Functions = {}));
})(TcHmi);
TcHmi.Functions.registerFunctionEx('ucButtonsImageJS1', 'TcHmi.Functions.Tc3Hmi_base', TcHmi.Functions.Tc3Hmi_base.ucButtonsImageJS1);
```

FIGURE 14. User control image color



```
// Keep these lines for a best effort IntelliSense of Visual Studio 2017 and higher.
/// <reference path="./../../Packages/Beckhoff.TwinCAT.HMI.Framework.12.758.8/runtimes/native1.12-tchmi/TcHmi.d.ts" />

(function (/** @type {globalThis.TcHmi} */ TcHmi) {
    var Functions;
    (function (/** @type {globalThis.TcHmi.Functions} */ Functions) {
        var Tc3Hmi_base;
        (function (Tc3Hmi_base) {
            function ucButtonsJS1(buttonID, IntButtonID, color) {

                if (buttonID == IntButtonID) {
                    color = {
                        "color": "rgba(231,231,231,1)"
                    }
                }
                else {
                    color = {
                        "color": "rgba(87,87,86,1)"

                    }
                }
                return color;
            }
            Tc3Hmi_base.ucButtonsJS1 = ucButtonsJS1;
        })(Tc3Hmi_base = Functions.Tc3Hmi_base || (Functions.Tc3Hmi_base = {}));
    })(Functions = TcHmi.Functions || (TcHmi.Functions = {}));
})(TcHmi);
TcHmi.Functions.registerFunctionEx('ucButtonsJS1', 'TcHmi.Functions.Tc3Hmi_base', TcHmi.Functions.Tc3Hmi_base.ucButtonsJS1);
```

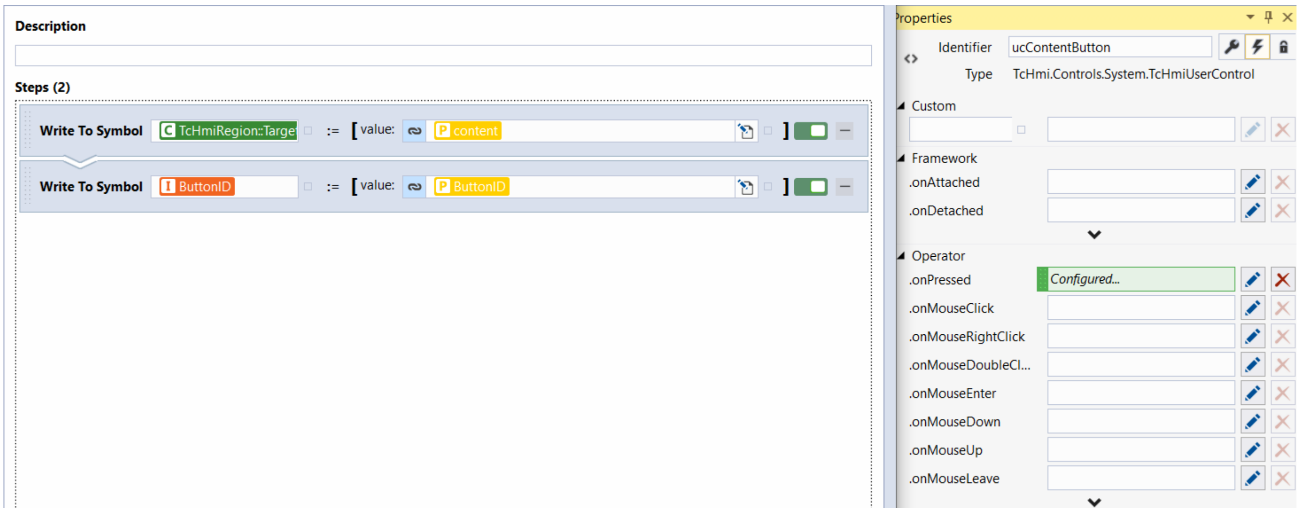
FIGURE 15. User control background color



```
// Keep these lines for a best effort IntelliSense of Visual Studio 2017 and higher.
/// <reference path="./../../Packages/Beckhoff.TwinCAT.HMI.Framework.12.758.8/runtimes/native1.12-tchmi/TcHmi.d.ts" />

(function (/** @type {globalThis.TcHmi} */ TcHmi) {
    var Functions;
    (function (/** @type {globalThis.TcHmi.Functions} */ Functions) {
        var Tc3Hmi_base;
        (function (Tc3Hmi_base) {
            function ucButtonTextJS1(buttonID, IntButtonID, color) {

                if (buttonID == IntButtonID) {
                    state = true;
                    color = {
                        "color": "rgba(87,87,86,1)"
                    }
                }
                else {
                    state = false;
                    color = {
                        "color": "rgba(231,231,231,1)"


                    }
                }
                return color;
            }
            Tc3Hmi_base.ucButtonTextJS1 = ucButtonTextJS1;
        })(Tc3Hmi_base = Functions.Tc3Hmi_base || (Functions.Tc3Hmi_base = {}));
    })(Functions = TcHmi.Functions || (TcHmi.Functions = {}));
})(TcHmi);
TcHmi.Functions.registerFunctionEx('ucButtonTextJS1', 'TcHmi.Functions.Tc3Hmi_base', TcHmi.Functions.Tc3Hmi_base.ucButtonTextJS1);
```

FIGURE 16. User control text color

Actions to the user controls can be added (Figure 17). When pressed, variable “symbol” is written to a Desktop.view’s region “target content”-parameter. This makes views to change when user control button is pressed. Also, user control’s “ButtonID” variable is written to the HMI’s internal symbol “ButtonID”. This makes JavaScript function if-statements to go true. Later when applying buttons, the id of each button can be defined.


Description

Steps (2)

Write To Symbol TcHmiRegion::Targe := [value: content]

Write To Symbol ButtonID := [value: ButtonID]

Properties

Identifier ucContentButton

Type TcHmi.Controls.System.TcHmiUserControl

Custom

Framework

.onAttached

.onDetached

Operator

.onPressed Configured...

.onMouseClick

.onMouseRightClick

.onMouseDoubleCl...

.onMouseEnter

.onMouseDown

.onMouseUp

.onMouseLeave

FIGURE 17. User control on pressed.

User controls can be added to a view just like all other controls. There are 7 buttons (Figure 18), one for each content made. These buttons do not have text or image until those are added.

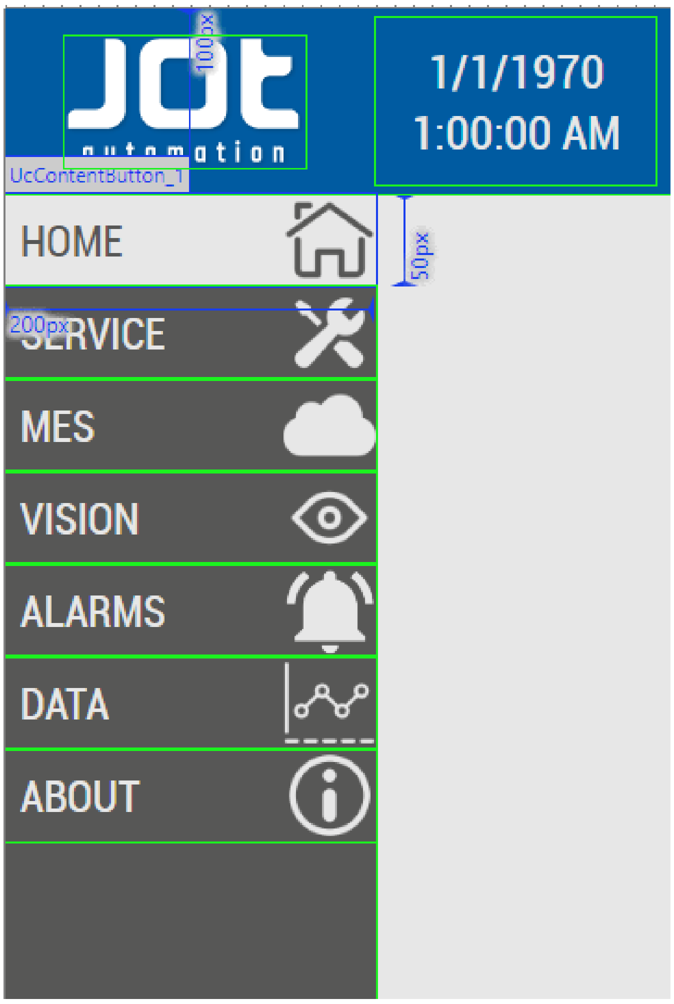
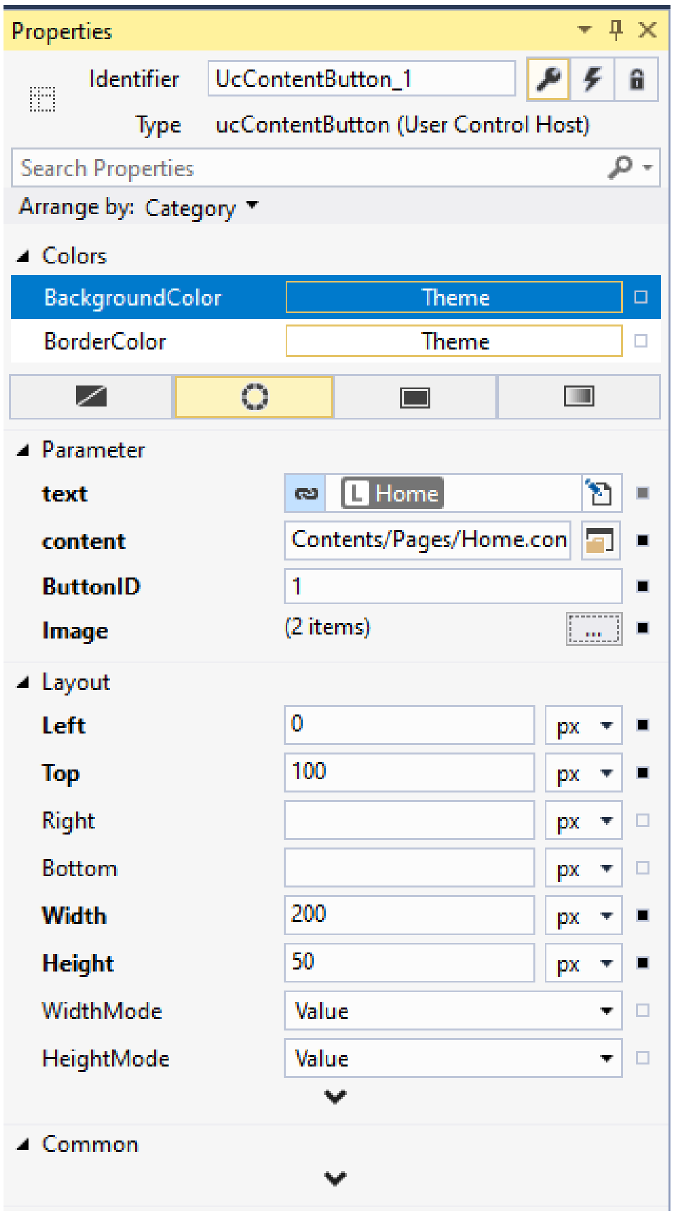


FIGURE 18. User controls in Desktop.view.

All the parameters in Figure 19 are used to give that button unique look. “Text” parameter gives text to text block inside user control. Target content of a region is chosen with “content” parameter. Parameter “ButtonID” is used to toggle just this button, so it needs to be different in all buttons. This button is first, so we use number one. Pressing … of parameter “Image”, the state image list can be made. Two images with same logo is used, but the color is different (Figure 20.). Value true links to a gray image and false links to a white image.



Properties

Identifier	UcContentButton_1
Type	ucContentButton (User Control Host)

Search Properties

Arrange by: Category

Colors

BackgroundColor	Theme
BorderColor	Theme

Parameter

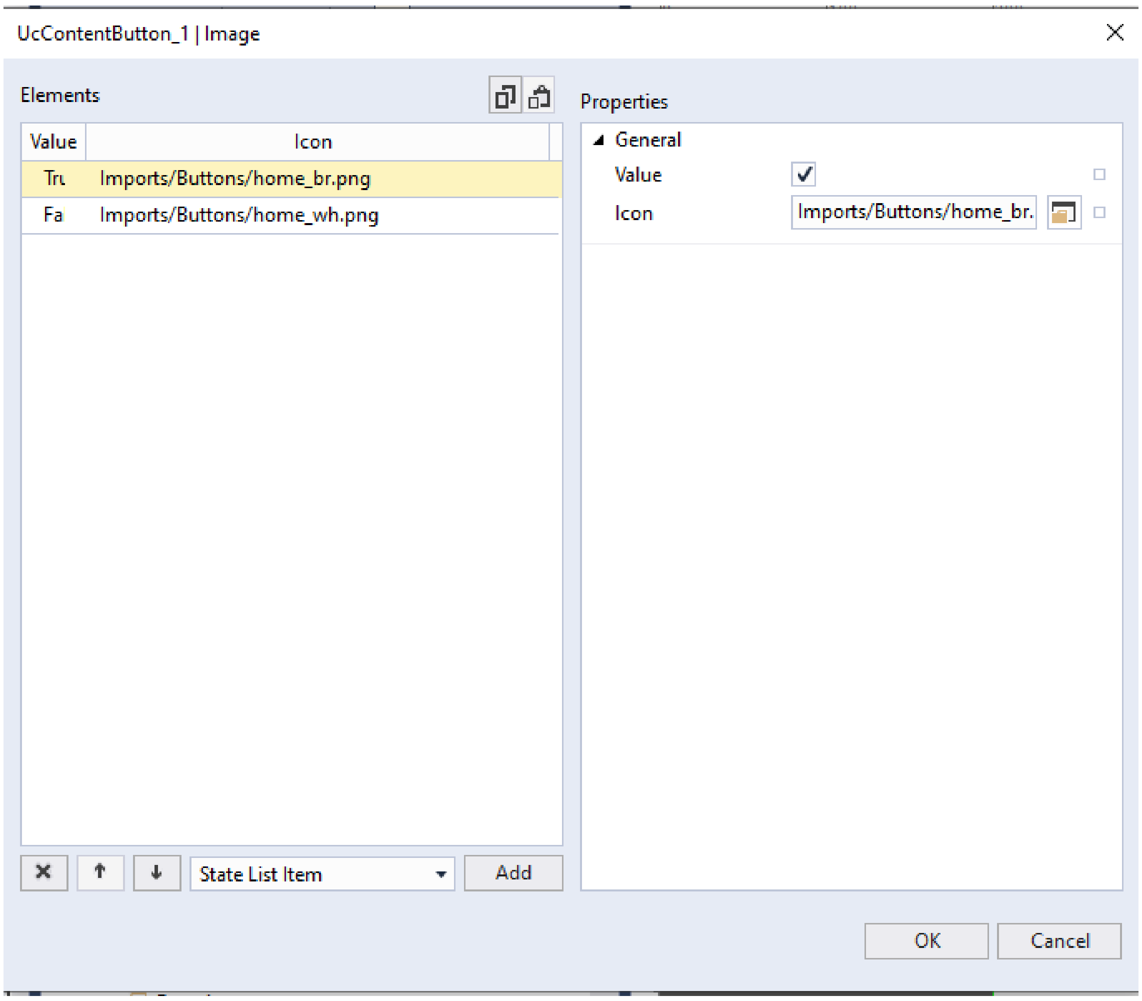
text	Home
content	Contents/Pages/Home.con
ButtonID	1
Image	(2 items)

Layout

Left	0	px
Top	100	px
Right		px
Bottom		px
Width	200	px
Height	50	px
WidthMode	Value	
HeightMode	Value	

Common

FIGURE 19. UcContentButton_1 (Home-button) properties.


UcContentButton_1 | Image

Elements

Value	Icon
Tru	Imports/Buttons/home_br.png
Fa	Imports/Buttons/home_wh.png

Properties

General

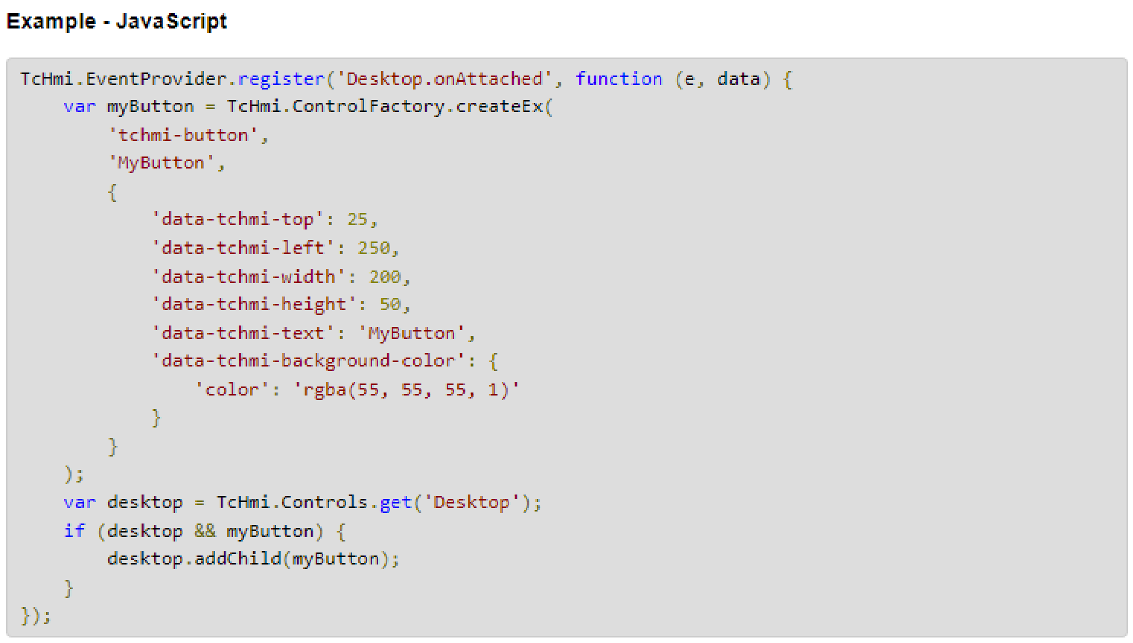
Value: ✓

Icon: Imports/Buttons/home_br.

FIGURE 20. UcContentButton_1 state list

4.3 Input/Output list

One of the key points with input and output lists was to reduce repeated tasks. A project with dozens of inputs, it would take much time to add individual indicators and names in list. To make IO-list to generate automatically, the JavaScript was used. The example in Figure 21 is from Beckhoff Infosys page.

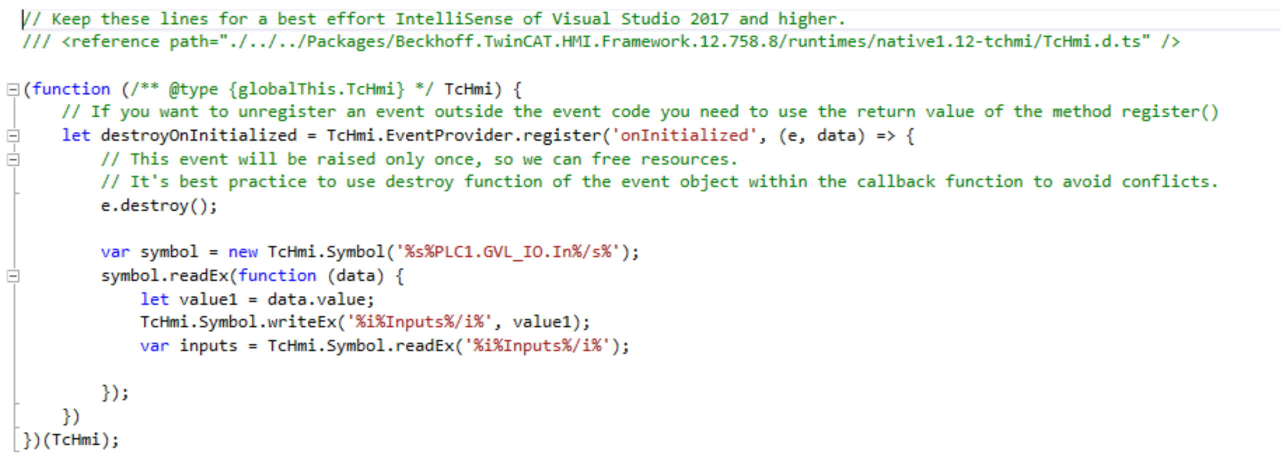


Example - JavaScript

```
TcHmi.EventProvider.register('Desktop.onAttached', function (e, data) {
    var myButton = TcHmi.ControlFactory.createEx(
        'tchmi-button',
        'MyButton',
        {
            'data-tchmi-top': 25,
            'data-tchmi-left': 250,
            'data-tchmi-width': 200,
            'data-tchmi-height': 50,
            'data-tchmi-text': 'MyButton',
            'data-tchmi-background-color': {
                'color': 'rgba(55, 55, 55, 1)'
            }
        }
    );
    var desktop = TcHmi.Controls.get('Desktop');
    if (desktop && myButton) {
        desktop.addChild(myButton);
    }
});
```

FIGURE 21. Example to add button with JavaScript (11).

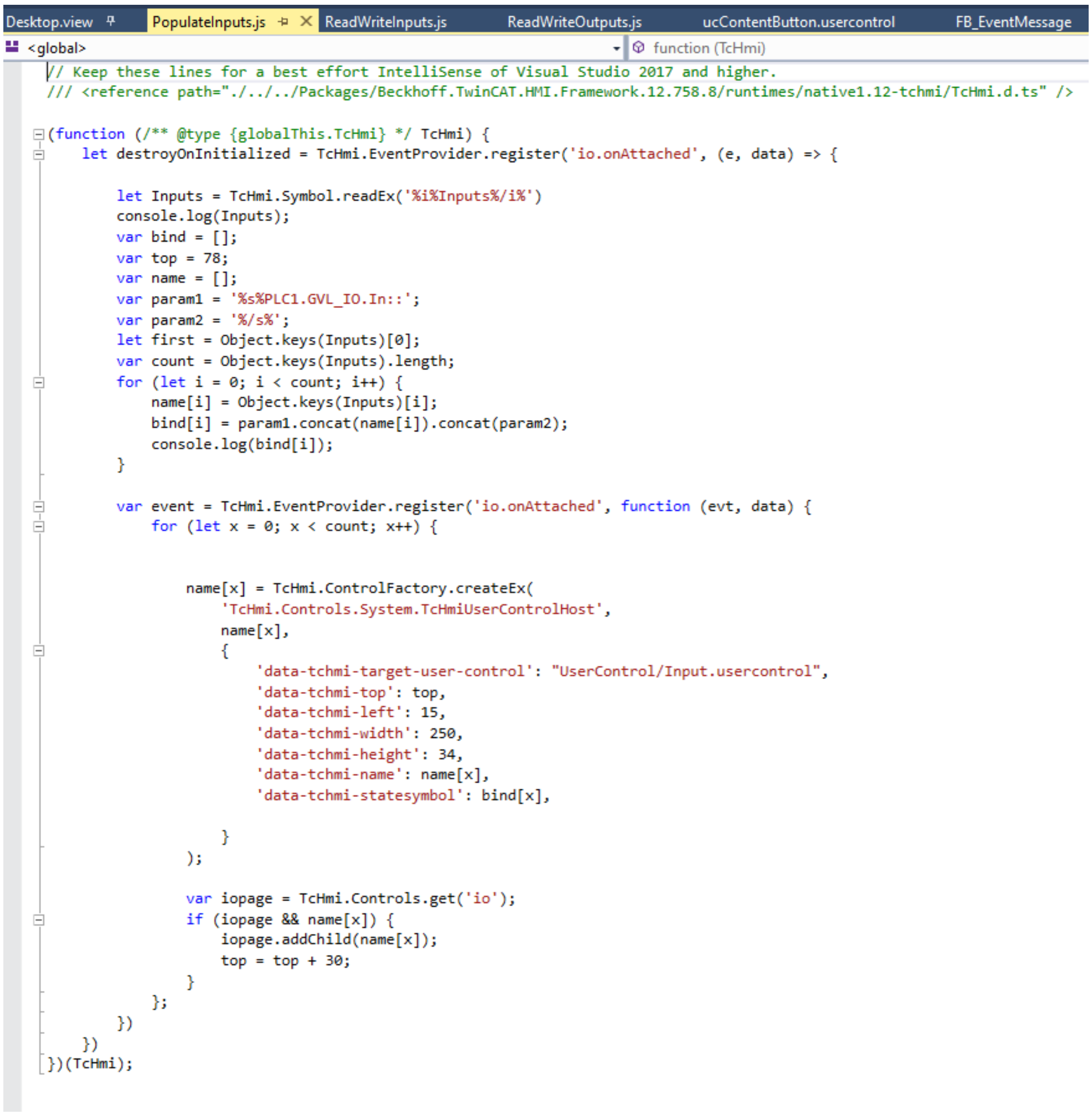
With many tests with JavaScript, the code generates as many user controls as there is inputs used in PLC. First code (Figure 22.) reads inputs from PLC and stores them in HMI project variable. Similar code to outputs were made and those were stored in own variable.



```
// Keep these lines for a best effort IntelliSense of Visual Studio 2017 and higher.
/// <reference path="./../../Packages/Beckhoff.TwinCAT.HMI.Framework.12.758.8/runtimes/native1.12-tchmi/TcHmi.d.ts" />

(function (/** @type {globalThis.TcHmi} */ TcHmi) {
    // If you want to unregister an event outside the event code you need to use the return value of the method register()
    let destroyOnInitialized = TcHmi.EventProvider.register('onInitialized', (e, data) => {
        // This event will be raised only once, so we can free resources.
        // It's best practice to use destroy function of the event object within the callback function to avoid conflicts.
        e.destroy();

        var symbol = new TcHmi.Symbol('%s%PLC1.GVL_IO.In%/s%');
        symbol.readEx(function (data) {
            let value1 = data.value;
            TcHmi.Symbol.writeEx('%i%Inputs%/i%', value1);
            var inputs = TcHmi.Symbol.readEx('%i%Inputs%/i%');

        });
    })
})(TcHmi);
```

FIGURE 22. Read and write inputs.

This codebehind JavaScript in Figure 23 is executed when io.content is attached to the region. At start code reads the stored value and stores it inside local array variable. Array values can be used to generate the user controls with for loop. Input name is stored in “name”- array and then the mapping variable “bind” is formed with adding “param1” and “param2” strings to it. Then code generates controls for each user control and adds them in io.content below of each other.



```
// Keep these lines for a best effort IntelliSense of Visual Studio 2017 and higher.
/// <reference path="./../../Packages/Beckhoff.TwinCAT.HMI.Framework.12.758.8/runtimes/native1.12-tchmi/TcHmi.d.ts" />

(function (/** @type {globalThis.TcHmi} */ TcHmi) {
    let destroyOnInitialized = TcHmi.EventProvider.register('io.onAttached', (e, data) => {

        let Inputs = TcHmi.Symbol.readEx('%i%Inputs%/i%')
        console.log(Inputs);
        var bind = [];
        var top = 78;
        var name = [];
        var param1 = '%s%PLC1.GVL_IO.In::';
        var param2 = '%/s%';
        let first = Object.keys(Inputs)[0];
        var count = Object.keys(Inputs).length;
        for (let i = 0; i < count; i++) {
            name[i] = Object.keys(Inputs)[i];
            bind[i] = param1.concat(name[i]).concat(param2);
            console.log(bind[i]);
        }

        var event = TcHmi.EventProvider.register('io.onAttached', function (evt, data) {
            for (let x = 0; x < count; x++) {


                name[x] = TcHmi.ControlFactory.createEx(
                    'TcHmi.Controls.System.TcHmiUserControlHost',
                    name[x],
                    {
                        'data-tchmi-target-user-control': "UserControl/Input.usercontrol",
                        'data-tchmi-top': top,
                        'data-tchmi-left': 15,
                        'data-tchmi-width': 250,
                        'data-tchmi-height': 34,
                        'data-tchmi-name': name[x],
                        'data-tchmi-statesymbol': bind[x],

                    }
                );

                var iopage = TcHmi.Controls.get('io');
                if (iopage && name[x]) {
                    iopage.addChild(name[x]);
                    top = top + 30;
                }
            };
        })
    })
})(TcHmi);
```

FIGURE 23. Populate inputs.

In Figure 24 there is the finished IO-list. Red circle in inputs display false-state and green circle true-state. Outputs have toggle buttons that shows current state ON/OFF and states can be changed by a user.

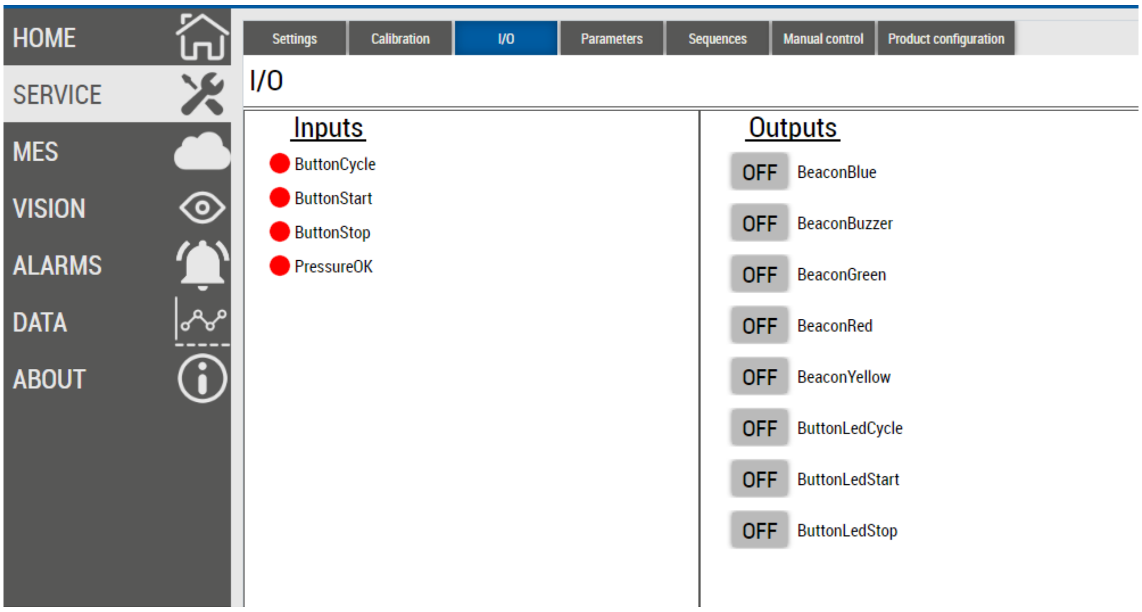


FIGURE 24. Input/Outputs lists

4.4 Displaying data

Displaying data in HMI needs chart and symbols, that are historized. Historizing does not include in basic HMI. It can be added as a NuGet Package. Project files has packages folder and right click opens list of options (Figure 25). There by selecting "Manage NuGet Packages" opens options for all the packages.

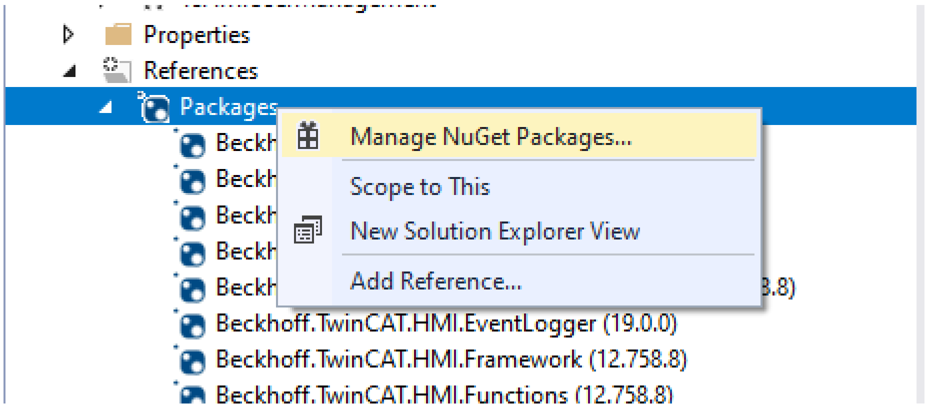
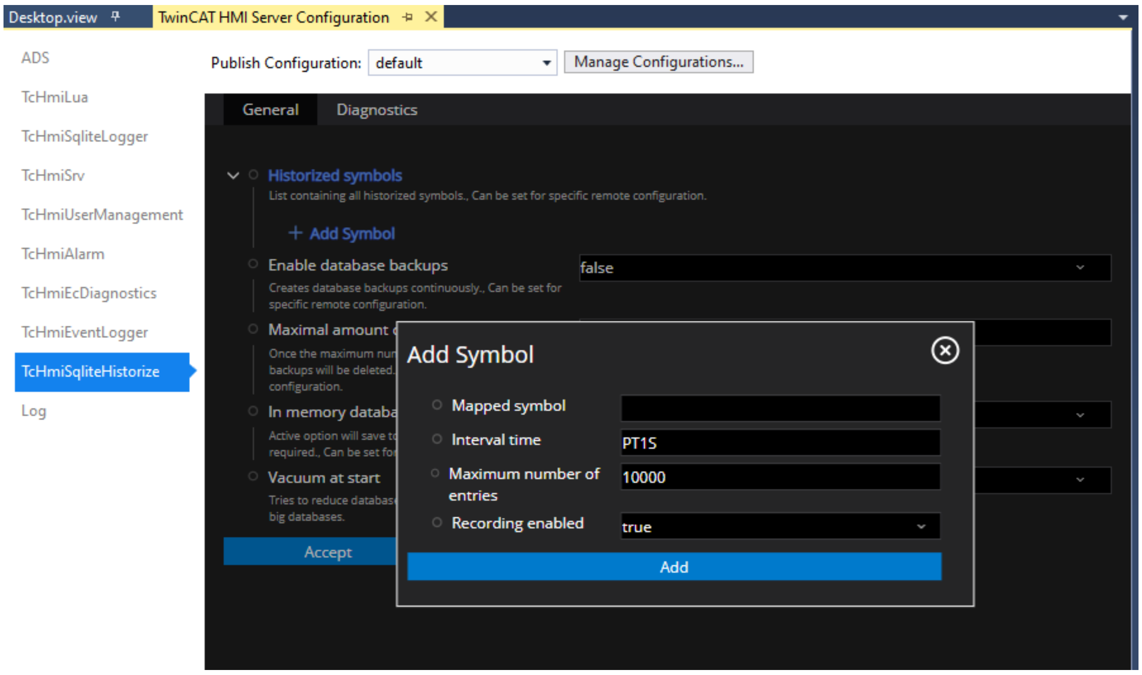

Properties
References
Packages
Beckh
Beckh
Beckh
Beckh
Beckh
Manage NuGet Packages...
Scope to This
New Solution Explorer View
Add Reference...
.8)
Beckhoff.TwinCAT.HMI.EventLogger (19.0.0)
Beckhoff.TwinCAT.HMI.Framework (12.758.8)
Beckhoff.TwinCAT.HMI.Functions (12.758.8)

FIGURE 25. Opening NuGet Packages

From Server configuration, historized symbols can be added (Figure 26). All the projects mapped symbols can be selected. There are options for the historizing interval and the maximum entries. Adding new symbol starts to historize it's states.


Publish Configuration: default

Historized symbols
List containing all historized symbols., Can be set for specific remote configuration.

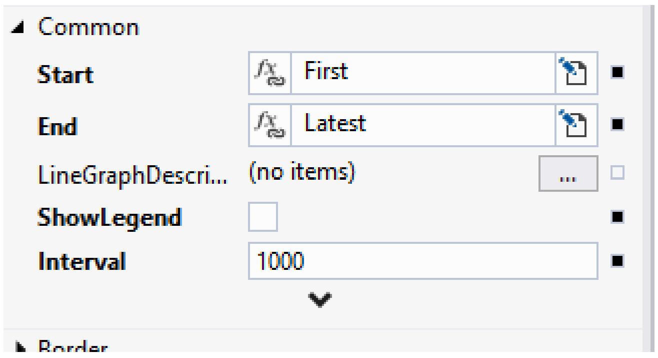
Enable database backups: false
Creates database backups continuously., Can be set for specific remote configuration.

Add Symbol

Mapped symbol	
Interval time	PT1S
Maximum number of entries	10000
Recording enabled	true

FIGURE 26. Add new historized symbol.

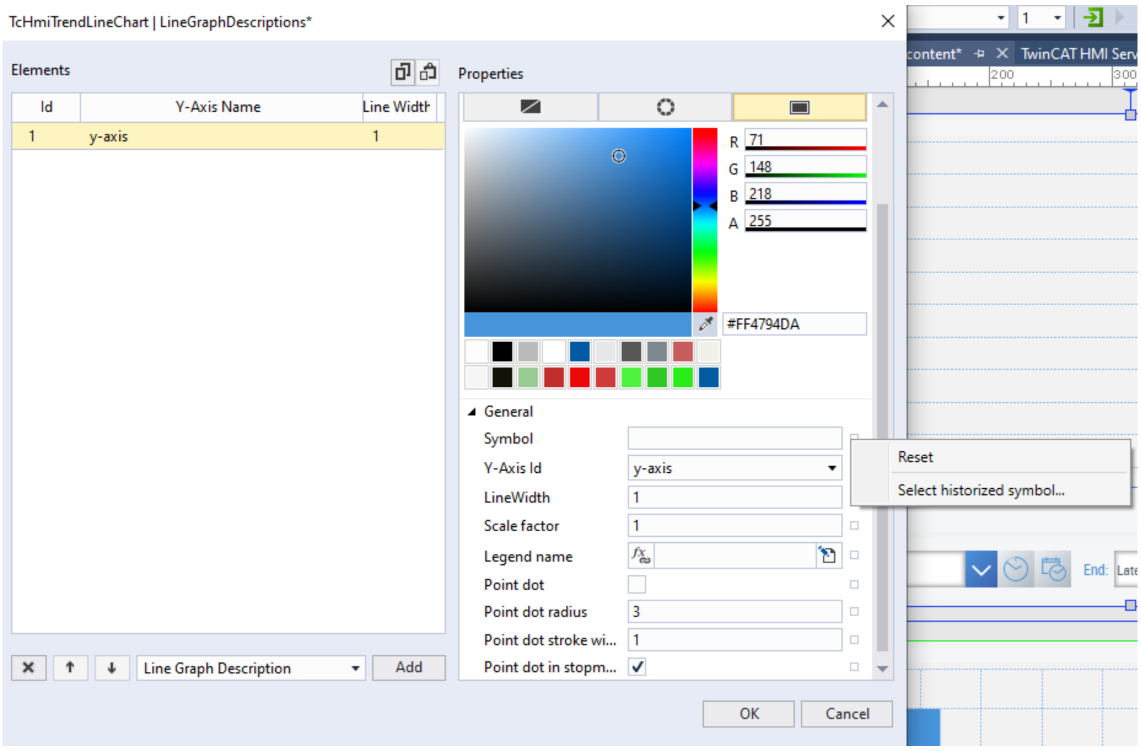
From HMI side there is needed to add chart. TcHmiTrendLineChart in HMI has properties shown in Figure 27. Line graph description can be added by pressing button with three dots.


Common

Start	First
End	Latest
LineGraphDescri...	(no items)
ShowLegend	
Interval	1000

FIGURE 27. TcHmiTrendLineChart properties

From the properties we can add historized symbol to the chart. Pressing white square next to symbol text opens option to select historized symbol (Figure 28). This opens list which show all the symbols that were added before. Now the y-axis has the historized symbol as a value and x-axis is counting time by default.


TcHmiTrendLineChart | LineGraphDescriptions*

Elements

Id	Y-Axis Name	Line Width
1	y-axis	1

Properties

R 71
G 148
B 218
A 255
#FF4794DA

General

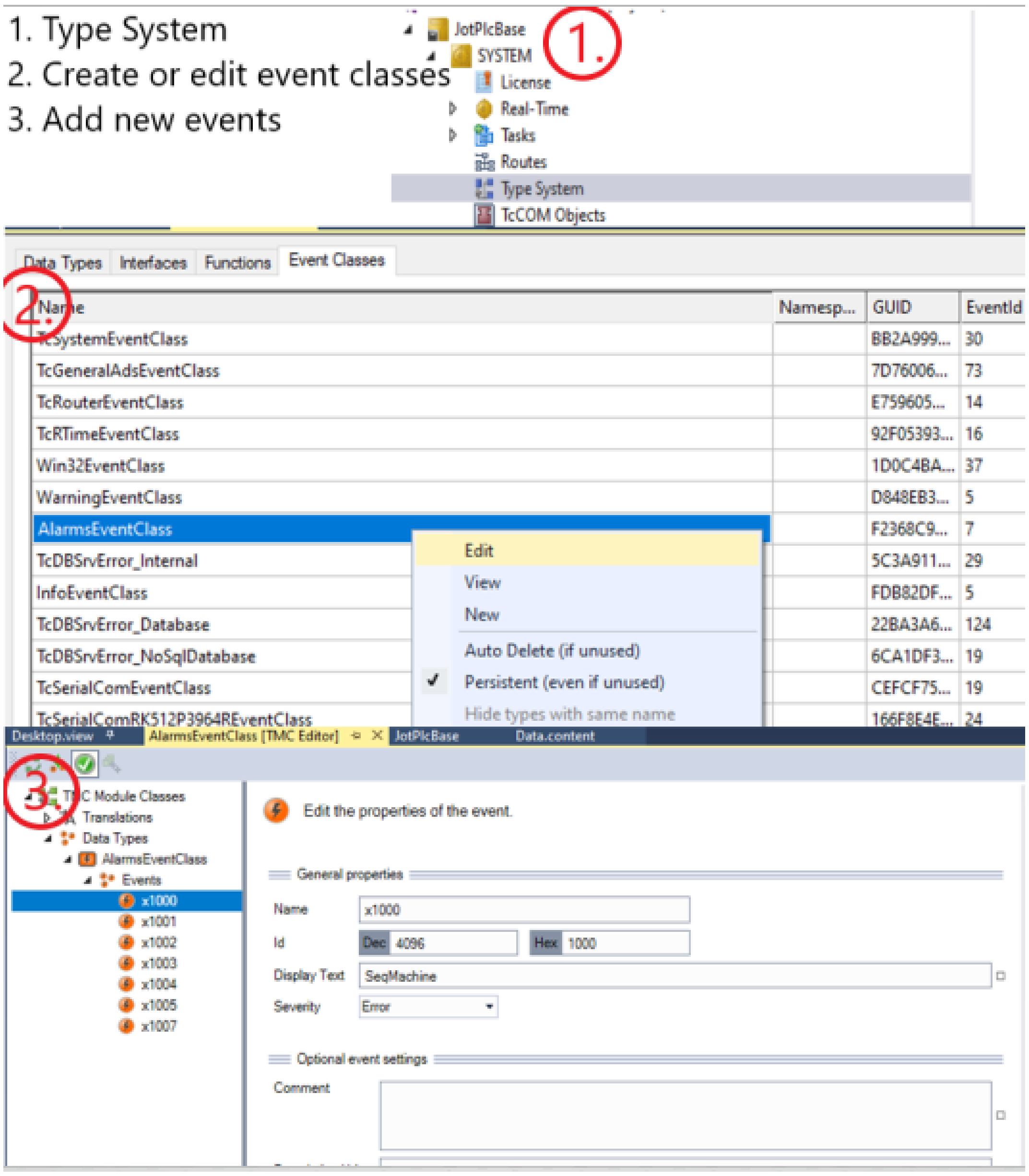
Symbol	
Y-Axis Id	y-axis
LineWidth	1
Scale factor	1
Legend name	
Point dot	
Point dot radius	3
Point dot stroke wi...	1
Point dot in stopm...	✓

FIGURE 28. *Y-axis properties*

4.5 Alarms system

Alarms can be added to the HMI with Eventlogger NuGet package. This allows to create messages, warnings, and errors to the HMI. Those can be used by operator or maintenance people to diagnose and control machine better.

There is only alarm-messages used in this project. Alarms can be created in type system settings. Figure 29 shows the steps to add new events. First step is to create new event class. Inside of the event class is the individual events. Those can be created and edited with given attributes. Event id is later used to get specific event.


1. Type System
2. Create or edit event classes
3. Add new events

Name	Namesp...	GUID	EventId
TcSystemEventClass		BB2A999...	30
TcGeneralAdsEventClass		7D76006...	73
TcRouterEventClass		E759605...	14
TcRTimeEventClass		92F05393...	16
Win32EventClass		1D0C4BA...	37
WarningEventClass		D848EB3...	5
AlarmsEventClass		F2368C9...	7
TcDBSrvError_Internal		5C3A911...	29
InfoEventClass		FDB82DF...	5
TcDBSrvError_Database		22BA3A6...	124
TcDBSrvError_NoSqlDatabase		6CA1DF3...	19
TcSerialComEventClass		CEFCF75...	19
TcSerialComRK512P3964REventClass		166F8E4E...	24

Edit the properties of the event.

General properties

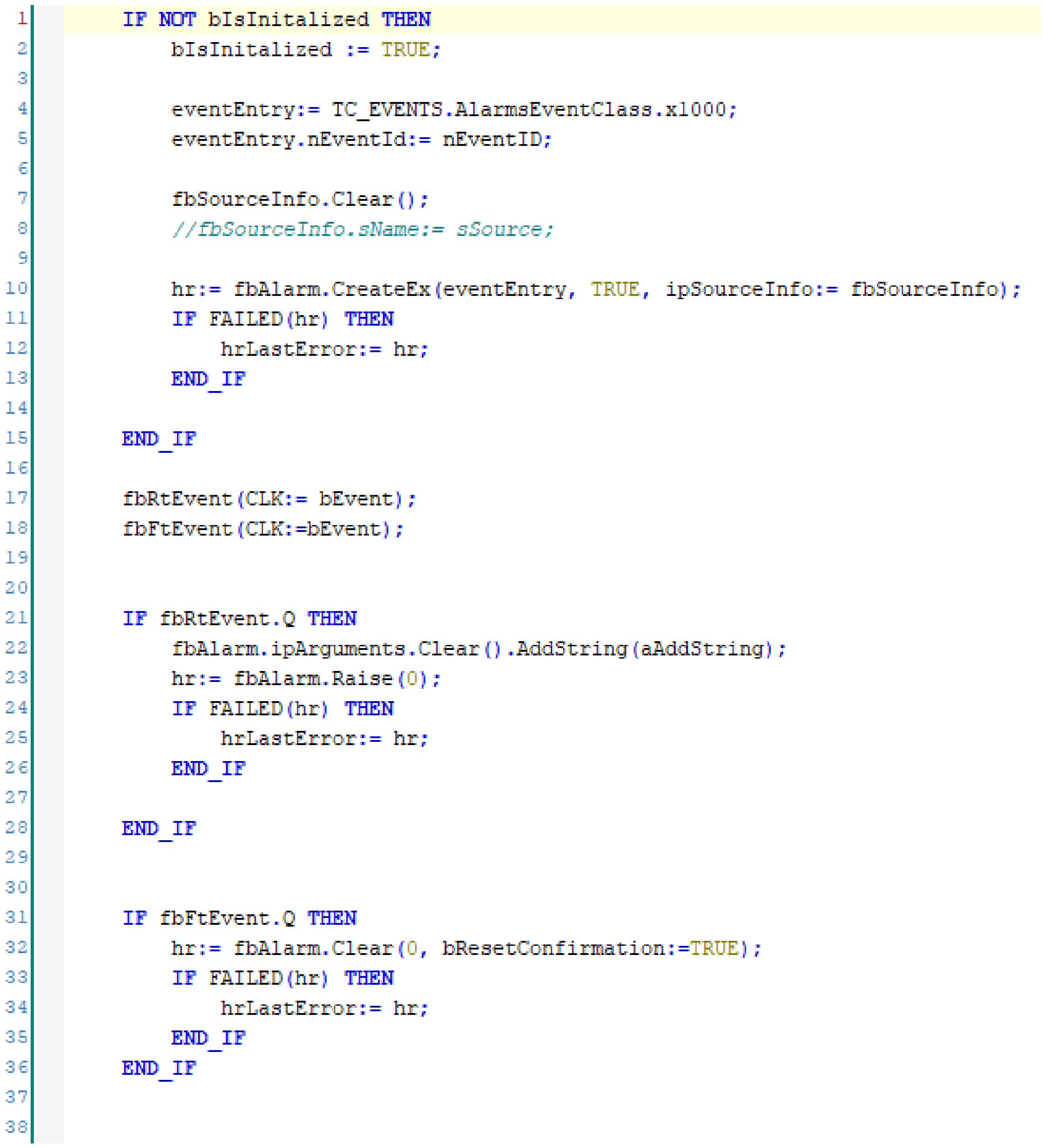
Name	x1000
Id	Dec 4096 Hex 1000
Display Text	SeqMachine
Severity	Error

Optional event settings

Comment

FIGURE 29. Adding new event

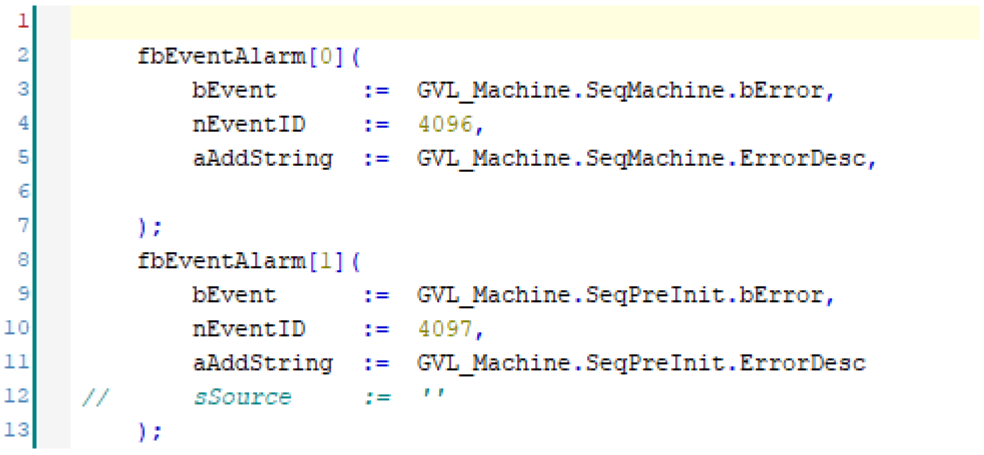
The PLC-project has its own errors already. The objective was to make then appear also in HMI. For this purpose, there is function block (Figure 30), that raises and clear the events. The function block is also used to add text in addition to alarm name.



```
IF NOT bIsInitalized THEN
    bIsInitalized := TRUE;

    eventEntry:= TC_EVENTS.AlarmsEventClass.x1000;
    eventEntry.nEventId:= nEventID;

    fbSourceInfo.Clear();
    //fbSourceInfo.sName:= sSource;

    hr:= fbAlarm.CreateEx(eventEntry, TRUE, ipSourceInfo:= fbSourceInfo);
    IF FAILED(hr) THEN
        hrLastError:= hr;
    END_IF

END_IF

fbRtEvent(CLK:= bEvent);
fbFtEvent(CLK:=bEvent);


IF fbRtEvent.Q THEN
    fbAlarm.ipArguments.Clear().AddString(aAddString);
    hr:= fbAlarm.Raise(0);
    IF FAILED(hr) THEN
        hrLastError:= hr;
    END_IF

END_IF


IF fbFtEvent.Q THEN
    hr:= fbAlarm.Clear(0, bResetConfirmation:=TRUE);
    IF FAILED(hr) THEN
        hrLastError:= hr;
    END_IF
END_IF

```

FIGURE 30. fbEventAlarm-function block

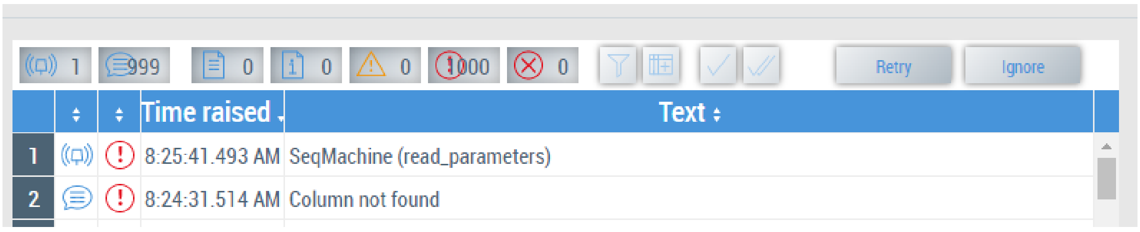
The program block was also created for the Eventlogger. There are multiple fbEventAlarm-function blocks. For every PLC-sequence there is one function block. In Figure 31 function blocks variables for the event raiser, event id, and the added string is determined. Variable “bEvent” is Boolean and is set true by PLC, when the “SeqMachine” is in error state. “nEventID” refers to the event id that was earlier determined in type system settings. To make error in HMI easier to understand, the “aAddString” variable is used to add PLC’s error description after the event text.



```
fbEventAlarm[0](
    bEvent      :=  GVL_Machine.SeqMachine.bError,
    nEventID    :=  4096,
    aAddString  :=  GVL_Machine.SeqMachine.ErrorDesc,

);
fbEventAlarm[1](
    bEvent      :=  GVL_Machine.SeqPreInit.bError,
    nEventID    :=  4097,
    aAddString  :=  GVL_Machine.SeqPreInit.ErrorDesc
//  sSource     :=  ''
);
```

FIGURE 31. Program for the Eventlogger

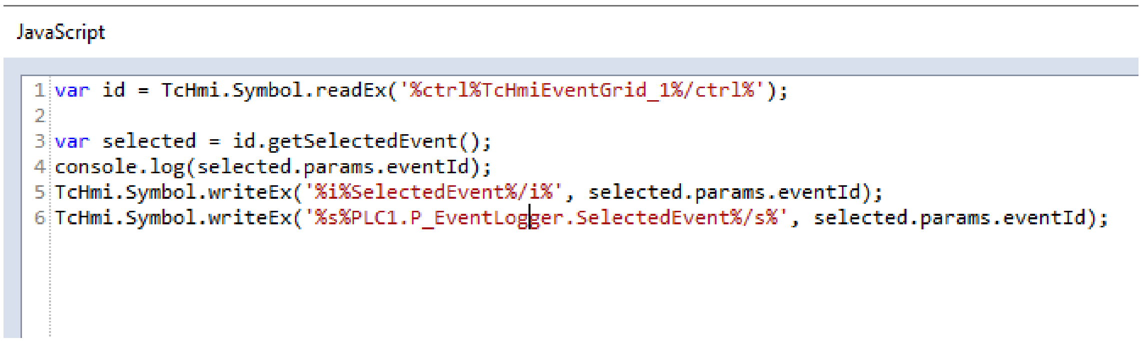
In Figure 32 there is alarm that was raised in PLC. Row 1 text column shows the error's source and error description next to it. Eventlogger also can be used to show PLC-errors. Row 2 shows PLC-error that was not raised with function block.



			Time raised	Text
1			8:25:41.493 AM	SeqMachine (read_parameters)
2			8:24:31.514 AM	Column not found

FIGURE 32. Eventlogger view in HMI

For future there is possibility to get event id to the PLC. This can be used to retry or ignore selected event. In Figure 33 there is example code in JavaScript, that writes event id to the PLC and HMI's internal symbol.



JavaScript

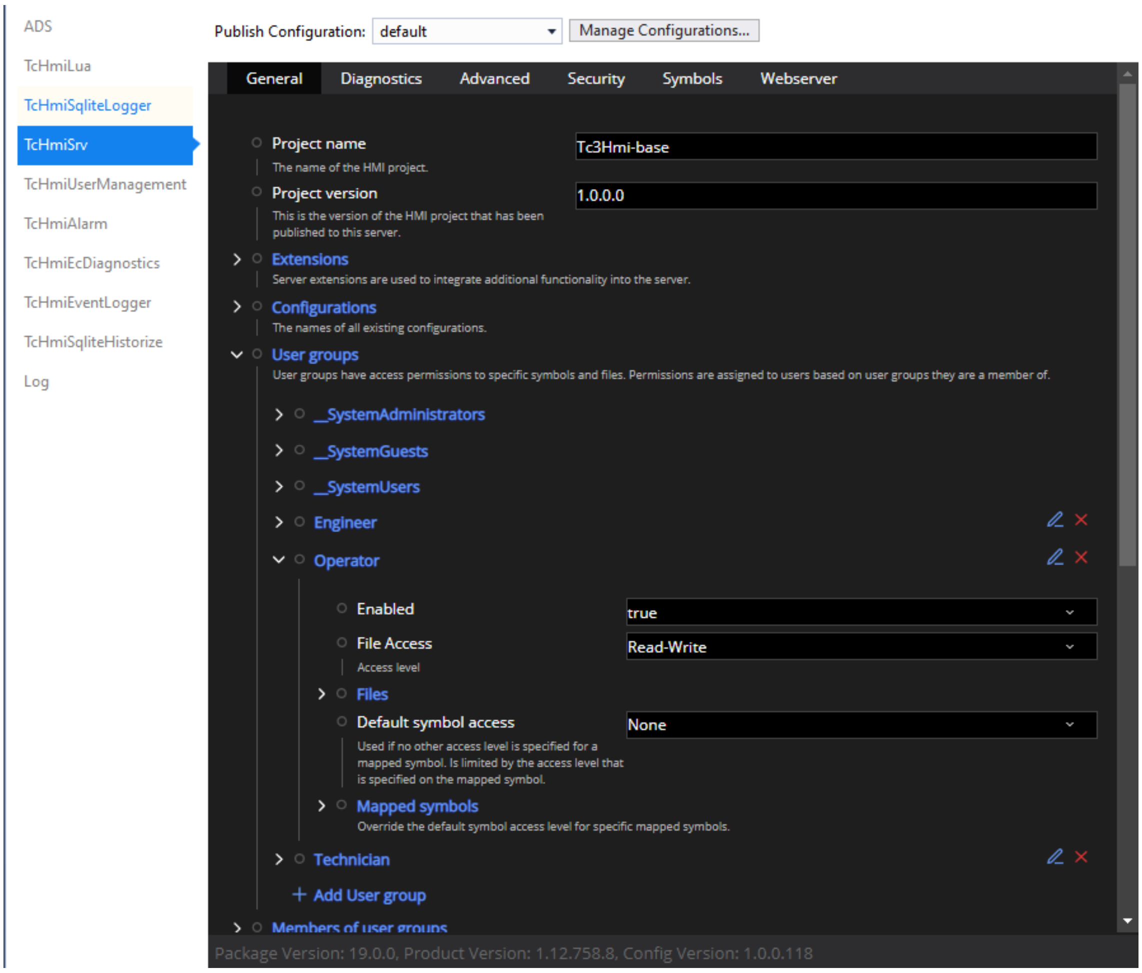
```javascript
var id = TcHmi.Symbol.readEx('%ctrl%TcHmiEventGrid_1%/ctrl%');

var selected = id.getSelectedEvent();
console.log(selected.params.eventId);
TcHmi.Symbol.writeEx('%i%SelectedEvent%/i%', selected.params.eventId);
TcHmi.Symbol.writeEx('%s%PLC1.P_EventLogger.SelectedEvent%/s%', selected.params.eventId);
```

FIGURE 33. JavaScript for writing selected event.

4.6 User management

User management is important option for most automation solutions when the machine requires user interactions. Developers debug tools or unsafe parameters need to be hidden for regular users.

HMI needs user management packet for this feature. It can be added as NuGet package. From server configuration, user groups can be created (Figure 34). There are system user groups that can be used, but for this HMI there are three new groups named Engineer, Operator, and Technician.


ADS
TcHmiLua
TcHmiSqliteLogger
TcHmiSrv
TcHmiUserManagement
TcHmiAlarm
TcHmiEcDiagnostics
TcHmiEventLogger
TcHmiSqliteHistorize
Log

Publish Configuration: default | Manage Configurations...

General | Diagnostics | Advanced | Security | Symbols | Webserver

Project name: Tc3Hmi-base
The name of the HMI project.

Project version: 1.0.0.0
This is the version of the HMI project that has been published to this server.

Extensions
Server extensions are used to integrate additional functionality into the server.

Configurations
The names of all existing configurations.

User groups
User groups have access permissions to specific symbols and files. Permissions are assigned to users based on user groups they are a member of.

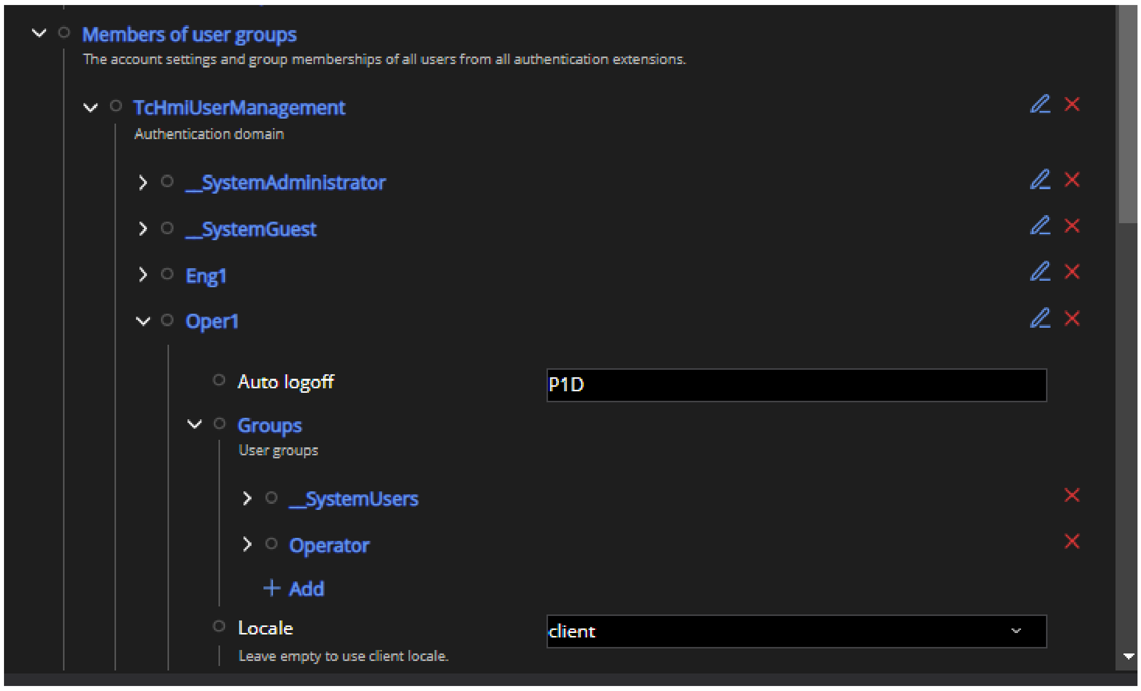
- __SystemAdministrators
- __SystemGuests
- __SystemUsers
- Engineer
- Operator
  - Enabled: true
  - File Access: Read-Write
    Access level
  - Files
  - Default symbol access: None
    Used if no other access level is specified for a mapped symbol. Is limited by the access level that is specified on the mapped symbol.
  - Mapped symbols
    Override the default symbol access level for specific mapped symbols.
- Technician
- + Add User group

Members of user groups

Package Version: 19.0.0, Product Version: 1.12.758.8, Config Version: 1.0.0.118

FIGURE 34. Server configuration

There is three new members and each of those have option to be part of groups made before. There is for example eng1-user that has rights to all three groups (engineer, technician, and operator). Oper1 belong only in group operator (Figure 35). This can be later used to limit operator access for certain controls.


Members of user groups
The account settings and group memberships of all users from all authentication extensions.

TcHmiUserManagement
Authentication domain

__SystemAdministrator
__SystemGuest
Eng1
Oper1

Auto logoff	P1D

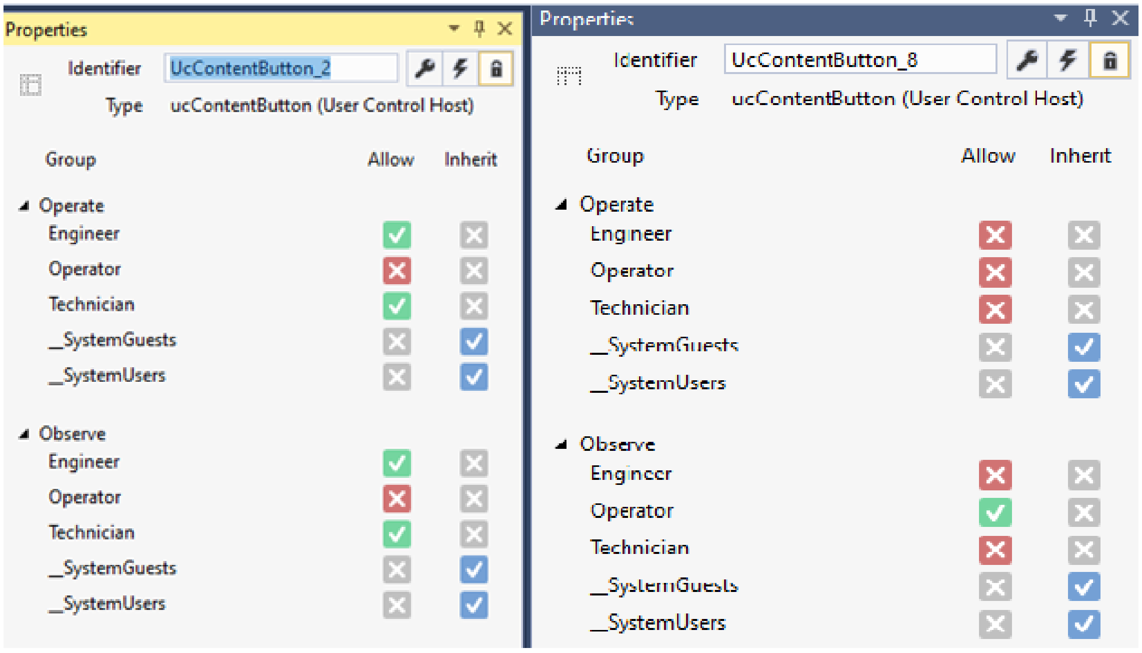
Groups
User groups
__SystemUsers
Operator
+ Add

Locale	client

Leave empty to use client locale.

FIGURE 35. Oper1 groups

All the elements in HMI have got a properties menu, that can be used to control user groups accesses. In Figure 36 there are properties of the two service buttons. There is another service button just for operators. UcContentButton_2 is set to be visible and operatable only for engineer and technician groups. However, UcContentButton_8 is only visible for operators, but it cannot be operated.


Properties
Identifier UcContentButton_2
Type ucContentButton (User Control Host)

Group	Allow	Inherit
Operate		
Engineer	✓	☒
Operator	✗	☒
Technician	✓	☒
__SystemGuests	☒	☑
__SystemUsers	☒	☑
Observe		
Engineer	✓	☒
Operator	✗	☒
Technician	✓	☒
__SystemGuests	☒	☑
__SystemUsers	☒	☑

Properties
Identifier UcContentButton_8
Type ucContentButton (User Control Host)

Group	Allow	Inherit
Operate		
Engineer	✗	☒
Operator	✗	☒
Technician	✗	☒
__SystemGuests	☒	☑
__SystemUsers	☒	☑
Observe		
Engineer	✗	☒
Operator	✓	☒
Technician	✗	☒
__SystemGuests	☒	☑
__SystemUsers	☒	☑

FIGURE 36. Service-button properties

As mentioned earlier there are two service buttons. Figure 37 left-hand side shows engineer's view and right-hand side shows operator's view. Engineer's button opens service page. Operator's button has no function behind it, and it does not open anything.

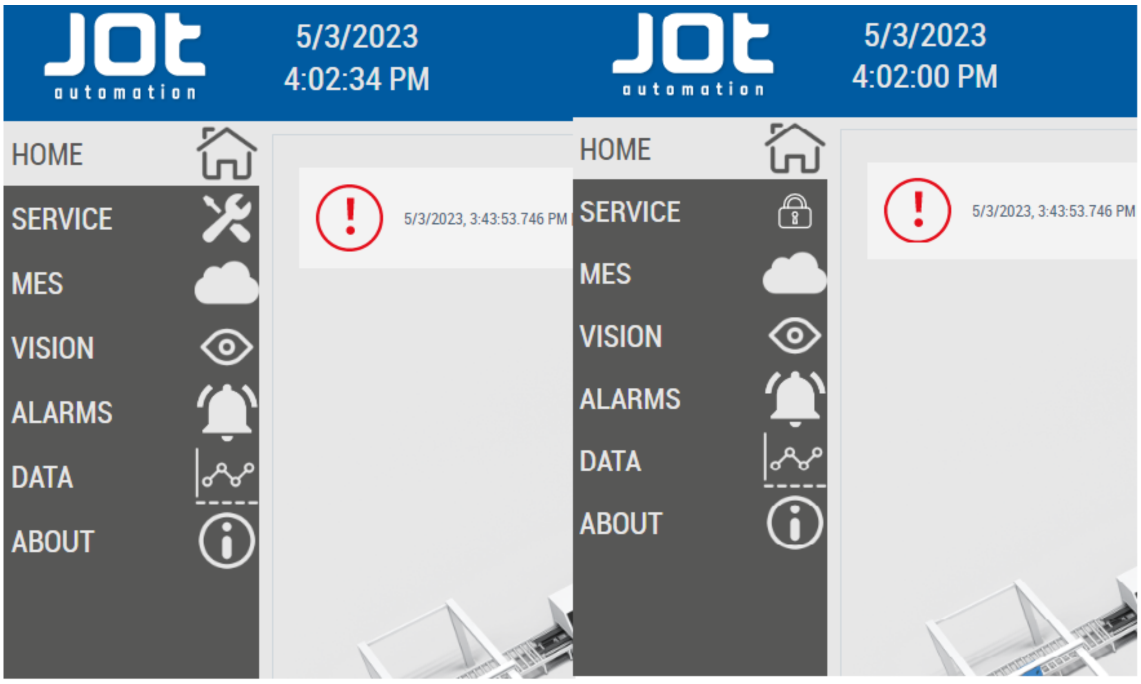
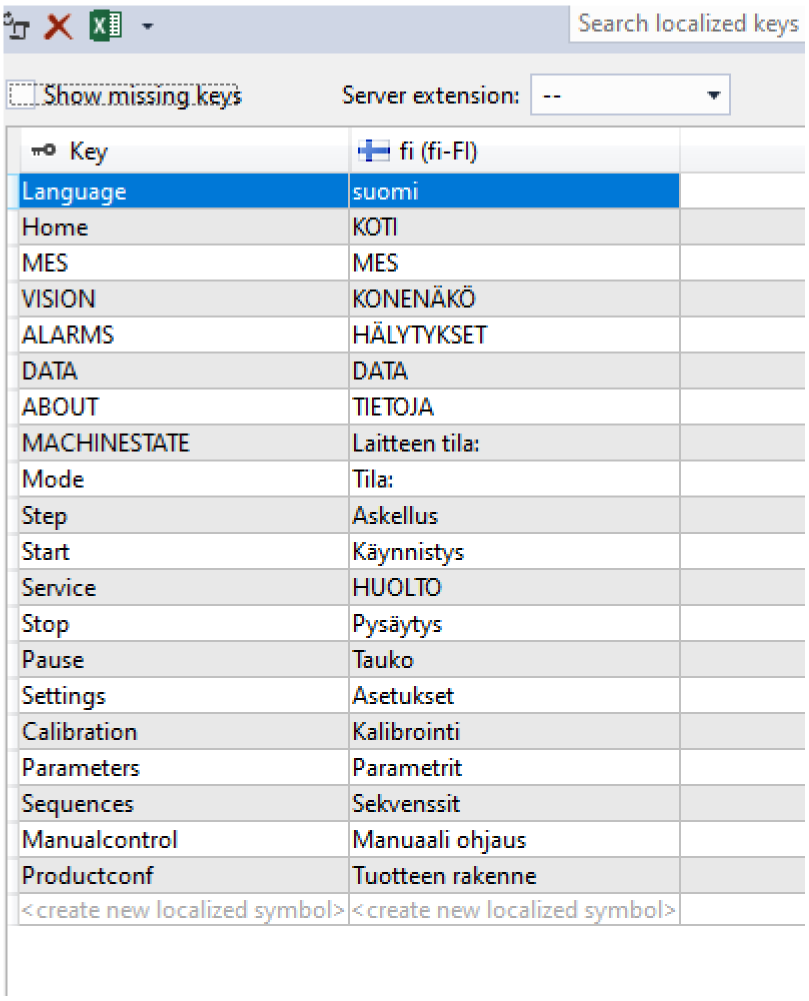


FIGURE 37. Service buttons

4.7 Localization

Localization can be used to translate all the texts of the HMI to a wanted language. In this project English and Finnish are used. Languages can be switched at any time.

Localization has its own folder by default and one language. Languages can be added with add new item feature used before. New language has two columns, one for keyword and one for the text (Figure 38). Keyword is same for both languages.

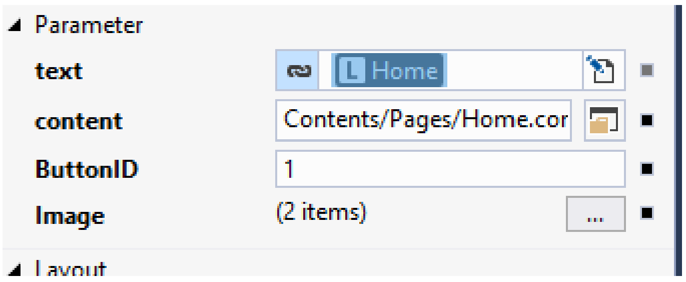

Search localized keys

Show missing keys Server extension: --

Key	fi (fi-FI)
Language	suomi
Home	KOTI
MES	MES
VISION	KONENÄKÖ
ALARMS	HÄLYTYKSET
DATA	DATA
ABOUT	TIETOJA
MACHINESTATE	Laitteen tila:
Mode	Tila:
Step	Askellus
Start	Käynnistys
Service	HUOLTO
Stop	Pysäytys
Pause	Tauko
Settings	Asetukset
Calibration	Kalibrointi
Parameters	Parametrit
Sequences	Sekvenssit
Manualcontrol	Manuaali ohjaus
Productconf	Tuotteen rakenne
<create new localized symbol>	<create new localized symbol>

FIGURE 38. Finnish translations

The keyword is used as symbol for the text. In Figure 39 there are parameters of the home-button. Text-parameter is localization-parameter, and it contains both languages.


Parameter

text	Home
content	Contents/Pages/Home.cor
ButtonID	1
Image	(2 items)

Layout

FIGURE 39. Home-button localization

Localization can be switched with TcHmiLozalizationSelect-control. Localization changes all the texts, that has localization used with it. Figure 40 shows navigation buttons in Finnish.

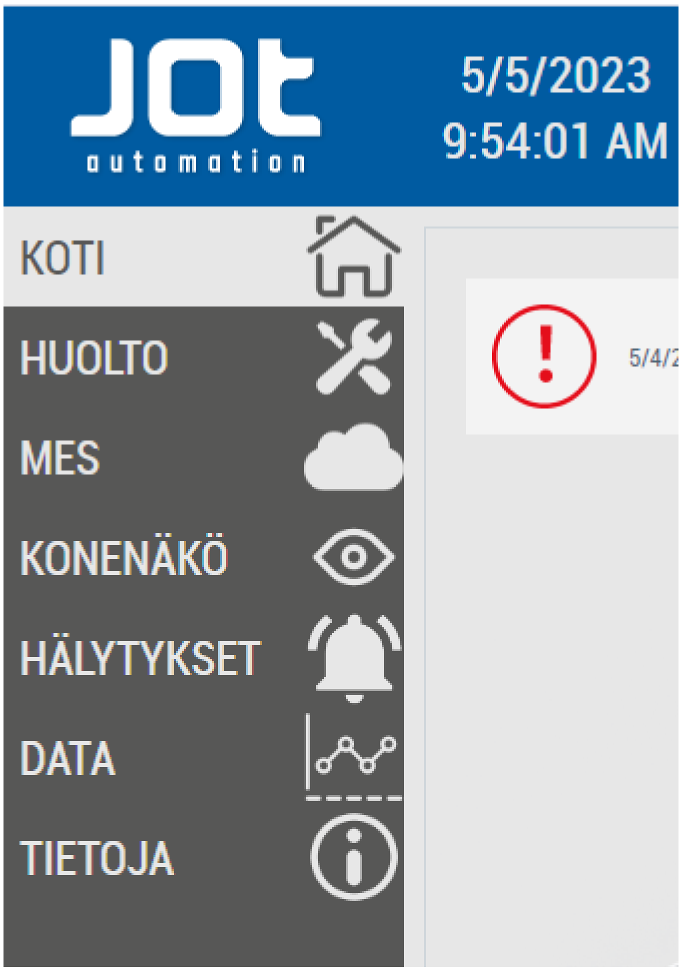


FIGURE 40. Navigation buttons in Finnish

5 SUMMARY

The goal of this thesis was to find out if the TwinCAT 3 HMI is suitable for the client’s automation solutions. The desired features for the customer were working alarms system, displaying data from SQL-server, user management, and automatically populating input/output-list. Also, additional features were to be researched.

The finished HMI has very responsive controls and it looks clear for the user. It looks mostly similar, and the basic functionalities work the same way with JOT’s current HMI. User management and the historized data were ready packages so those did not take a long time to implement. HMI’s alarms system needed some PLC code, and it needed some tests before I got it working. The toolbox had most of the needed controls ready, but those need some knowledge to get work wanted way. Beckhoff’s information system helped a lot with the controls parameters.

More challenging part was the IO-list. Client wanted it to automatically populate from PLC’s input/output variables. I was to learn JavaScript for that purpose. Beckhoff information system had some info of the JavaScript part, but I also needed to use other sources when learning. Knowing the basics of the Python programming language helped me a lot. IO-list came out good and can be further developed.

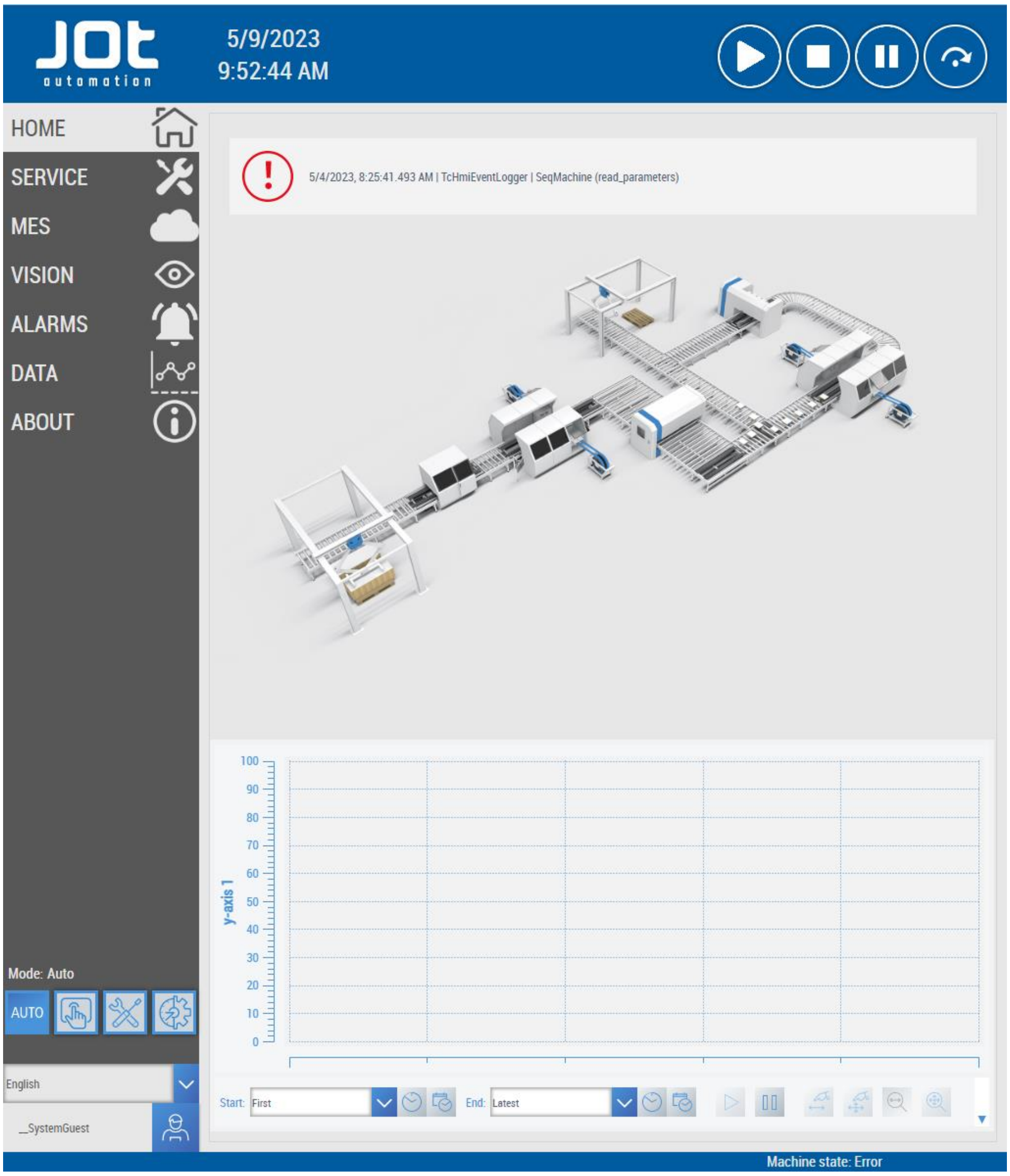
Getting data from SQL-server were found possible, but it was left out of the project. I would have needed more information about SQL-servers. Some PLC code were also needed, and it would have taken too much time. This is something that can be later implemented in the HMI.

TwinCAT 3 HMI supports HTML code in control called HTML host. It can be used to create many features. The extensions for the HMI can be made with C++ and .NET programming. This makes the HMI very customizable. There is also package for vision. I think all the features from JOT’s current HMI can be made with either of these.

To release the HMI for the customer, TwinCAT 3 HMI Server requires licence to work more than seven days. The TF2000-server licence includes a client connection and a target connection. More client connections can be applied with target pack licences. SQL-database, vision and extensions software packs requires own licences. Most probably JOT’s projects require more than just TF2000-licence.

REFERENCES

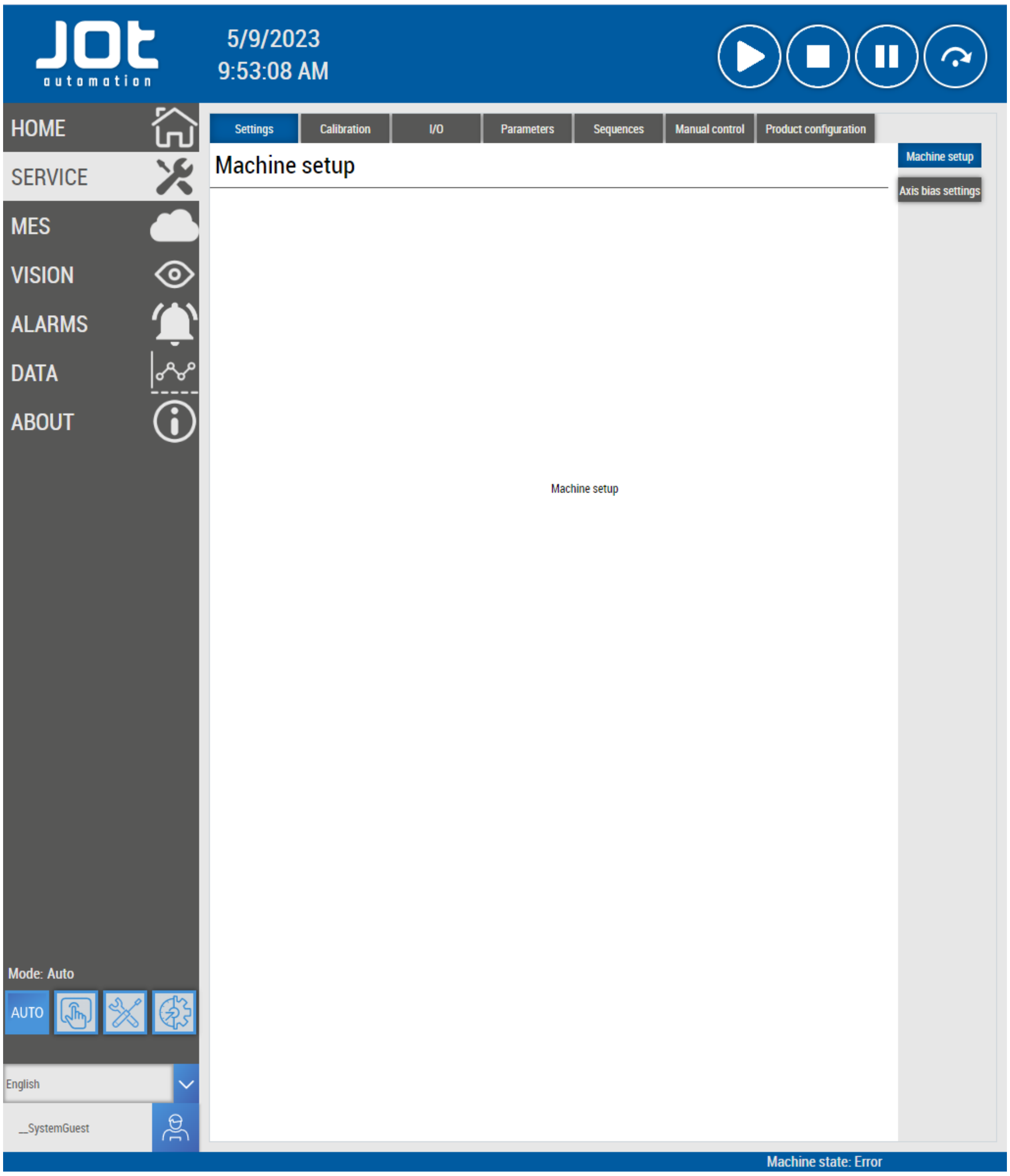
1. Jot Automation Ltd.. About us. Search date 9.3.2023. https://www.linkedin.com/company/jot-automation
2. Jot Automation Ltd.. Products. Search date 9.3.2023. https://www.jotautomation.com/products
3. Beckhoff automation. TwinCAT automation software. Search date 22.2.2023. https://www.beckhoff.com/fi-fi/products/automation/twincat/
4. Beckhoff automation. TE2000 | TwinCAT 3 HMI Engineering. Search date 22.2.2023. https://www.beckhoff.com/fi-fi/products/automation/twincat/texxxx-twincat-3-engineering/te2000.html
5. Beckhoff automation. TwinCAT HMI. Search date 22.2.2023. https://www.beckhoff.com/fi-fi/products/automation/twincat-3-hmi/
6. Inductive Automation LLC. What is HMI?. Search date 22.2.2023. https://www.inductiveautomation.com/resources/article/what-is-hmi
7. Jordana A. What Is JavaScript? A Basic Introduction to JS for Beginners. Search date 10.5.2023. https://www.hostinger.com/tutorials/what-is-javascript
8. Beckhoff automation. TF2000 | TwinCAT 3 HMI Server. Search date 10.5.2023. https://www.beckhoff.com/en-en/products/automation/twincat/tfxxxx-twincat-3-functions/tf2xxx-tc3-hmi/tf2000.html
9. Beckhoff automation. Beckhoff information system. TcHmiRegion. Search date 29.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/te2000_tc3_hmi_engineering/3845323019.html&id=
10. Beckhoff automation. Beckhoff information system. Content. Search date 29.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/te2000_tc3_hmi_engineering/3845328139.html&id=
11. Beckhoff automation. Beckhoff information system. createEx. Search date 30.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/te2000_tc3_hmi_engineering/4705724555.html&id=



5/4/2023, 8:25:41.493 AM | TcHmiEventLogger | SeqMachine (read_parameters)

Mode: Auto

Machine state: Error

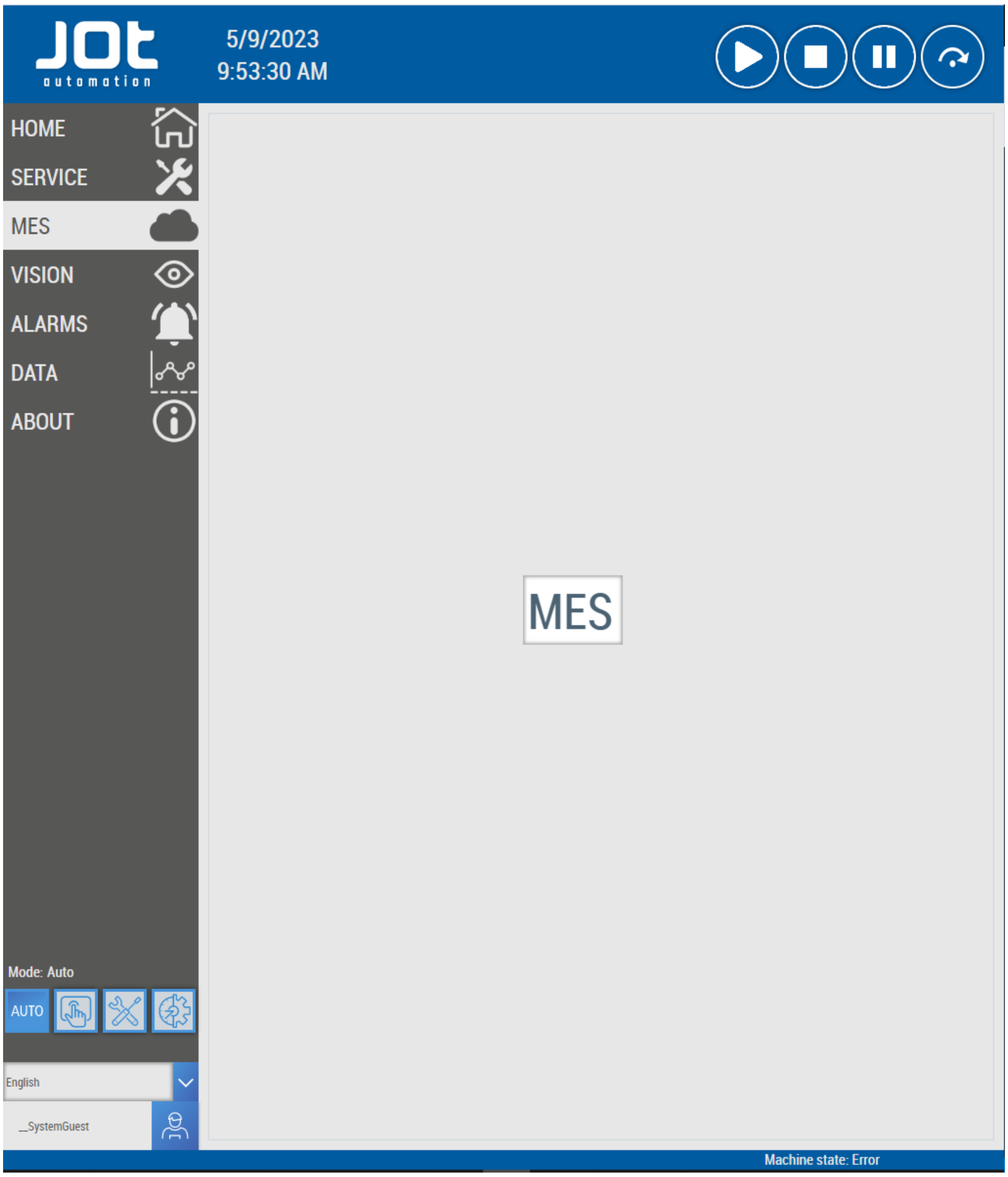


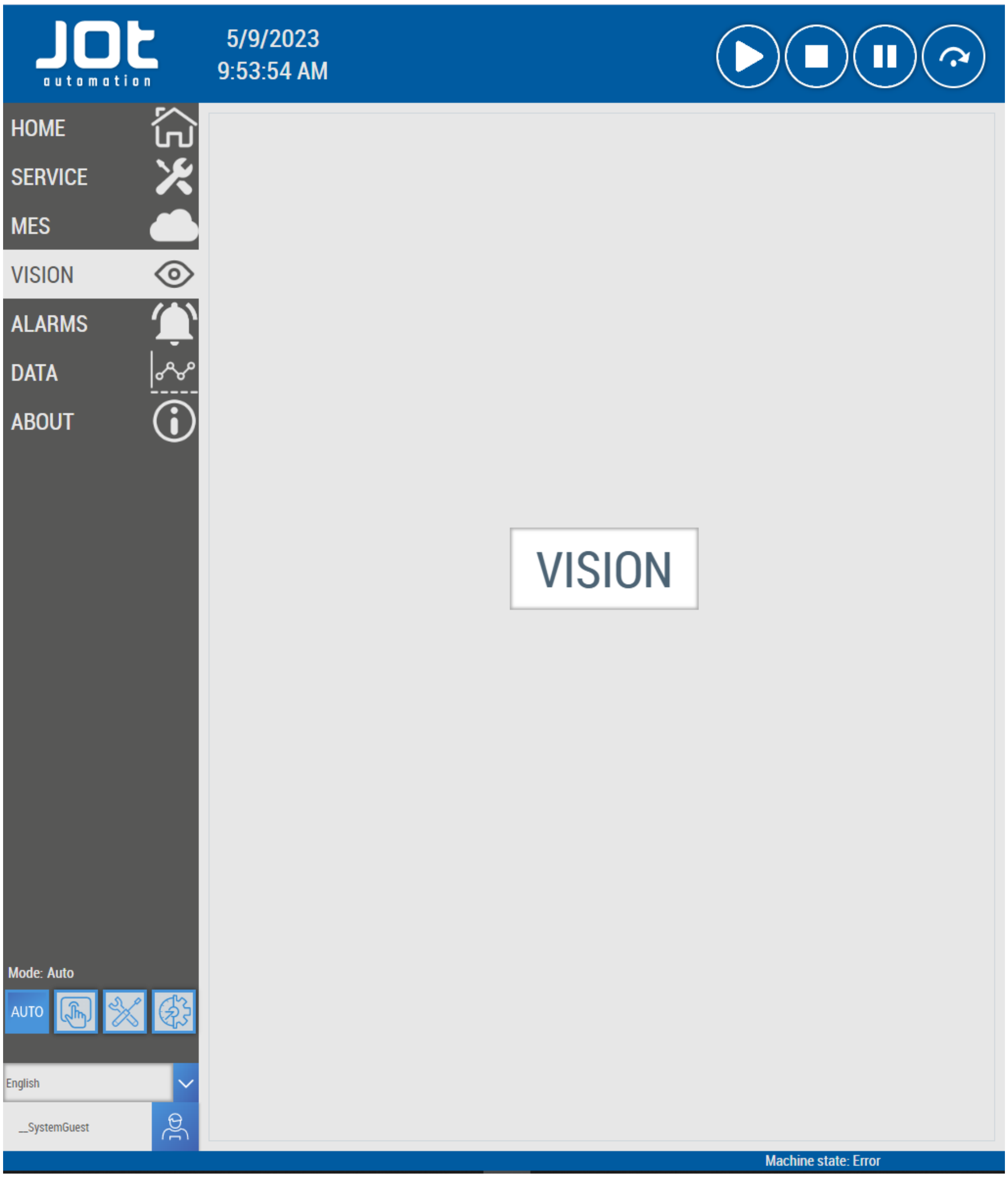
Machine setup

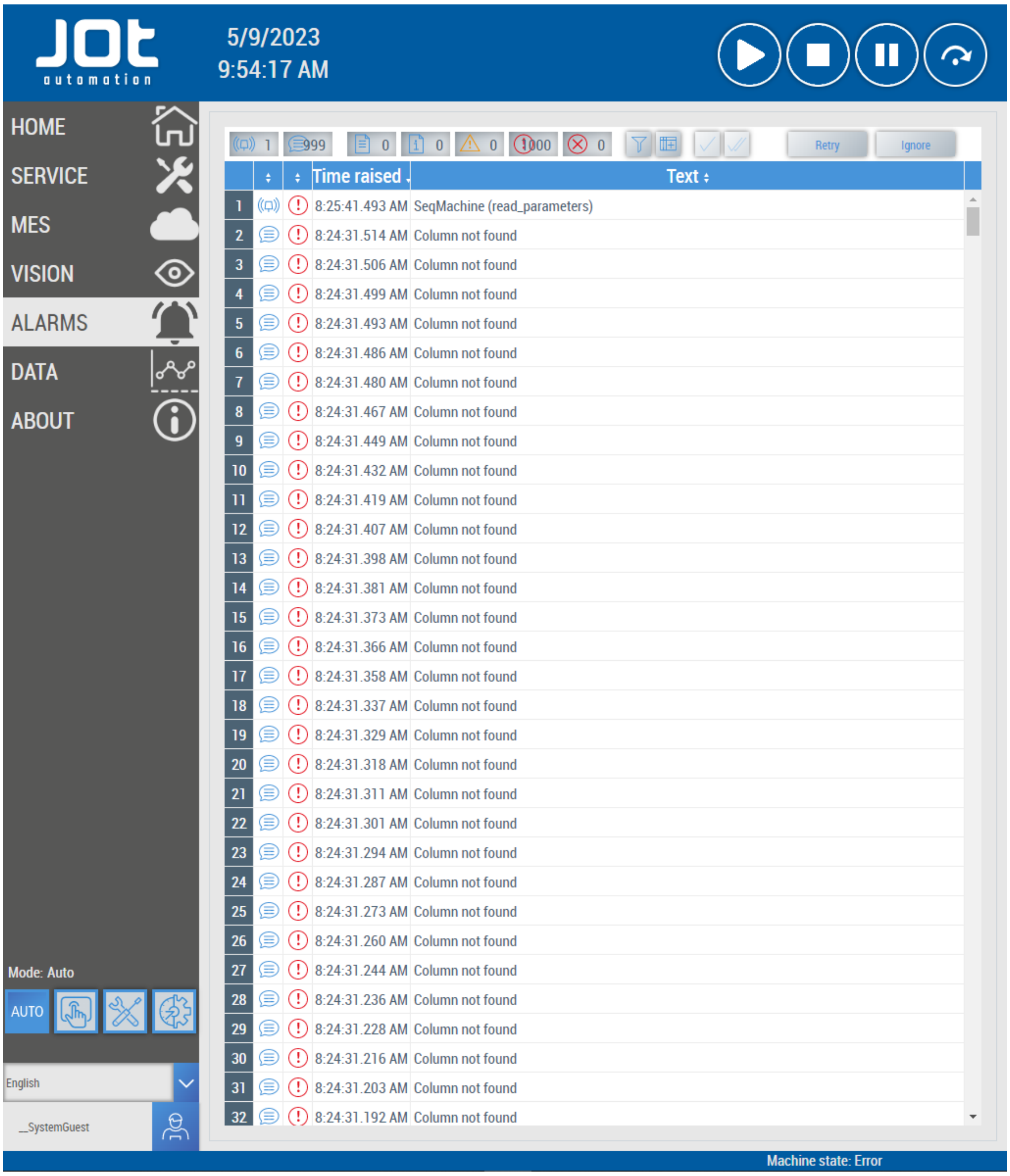
Machine setup

Mode: Auto

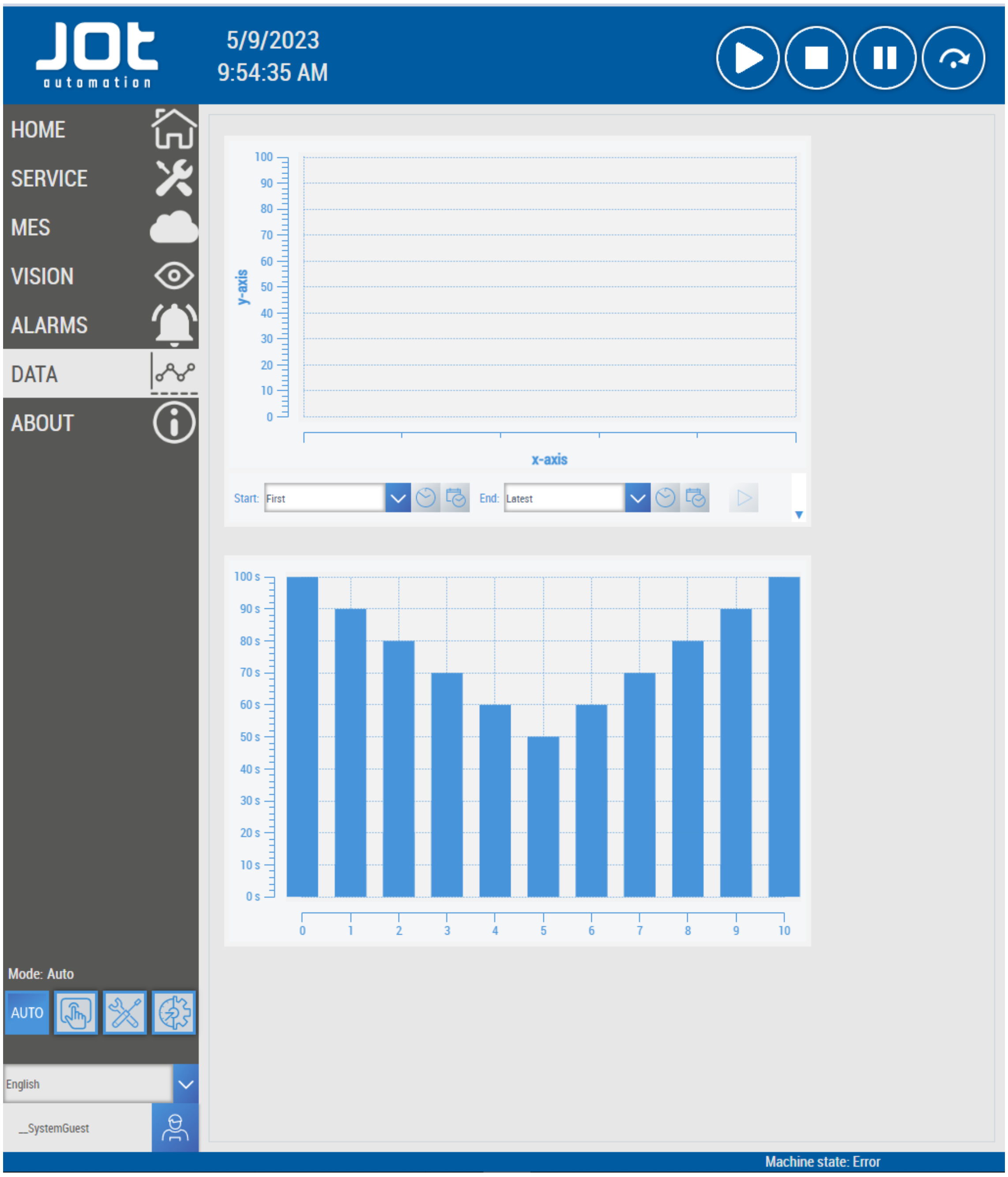
Machine state: Error

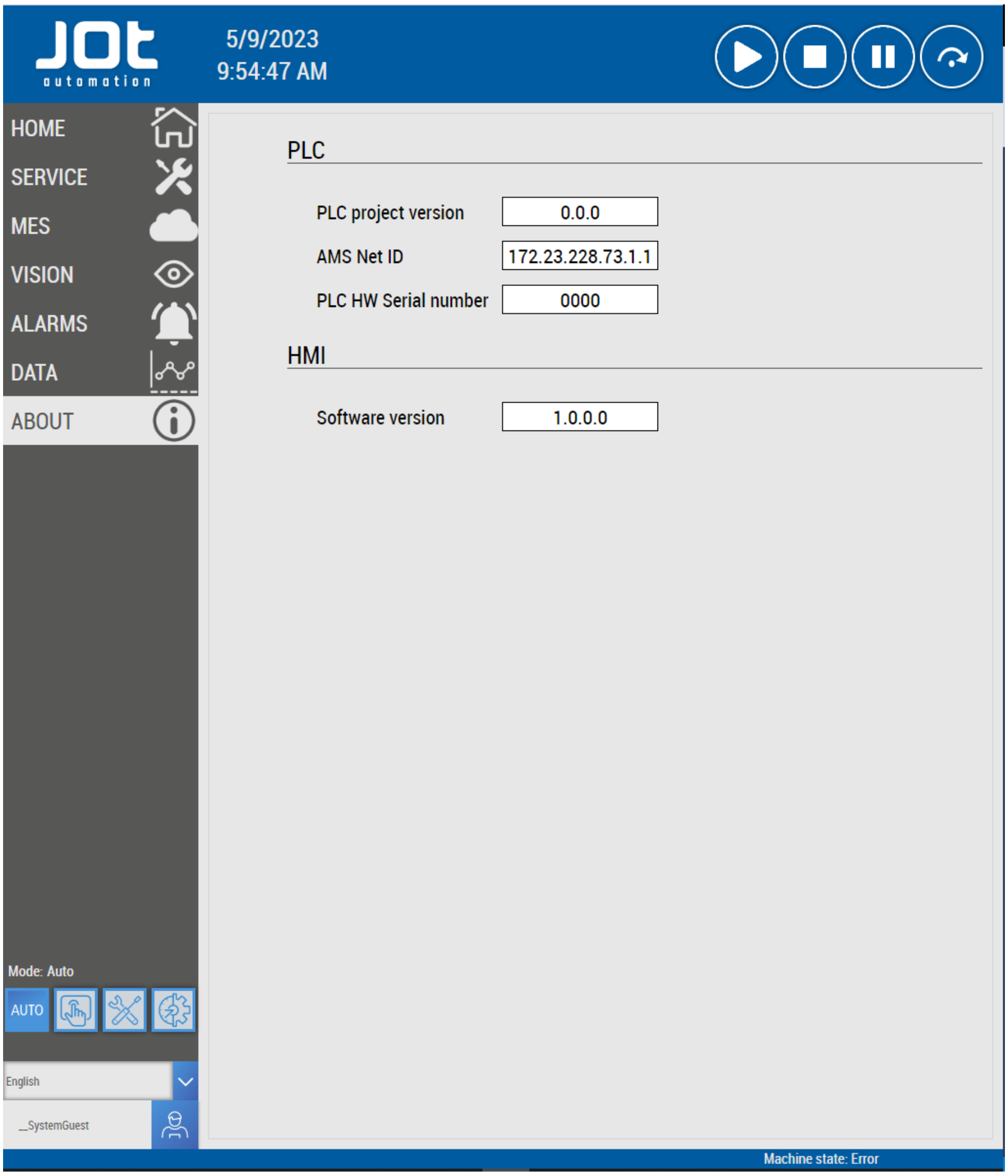

5/9/2023
9:53:30 AM
HOME
SERVICE
MES
VISION
ALARMS
DATA
ABOUT
MES
Mode: Auto
AUTO
English
__SystemGuest
Machine state: Error


5/9/2023
9:53:54 AM
HOME
SERVICE
MES
VISION
ALARMS
DATA
ABOUT
VISION
Mode: Auto
AUTO
English
__SystemGuest
Machine state: Error



			Time raised	Text
1	((□))	!	8:25:41.493 AM	SeqMachine (read_parameters)
2	≡	!	8:24:31.514 AM	Column not found
3	≡	!	8:24:31.506 AM	Column not found
4	≡	!	8:24:31.499 AM	Column not found
5	≡	!	8:24:31.493 AM	Column not found
6	≡	!	8:24:31.486 AM	Column not found
7	≡	!	8:24:31.480 AM	Column not found
8	≡	!	8:24:31.467 AM	Column not found
9	≡	!	8:24:31.449 AM	Column not found
10	≡	!	8:24:31.432 AM	Column not found
11	≡	!	8:24:31.419 AM	Column not found
12	≡	!	8:24:31.407 AM	Column not found
13	≡	!	8:24:31.398 AM	Column not found
14	≡	!	8:24:31.381 AM	Column not found
15	≡	!	8:24:31.373 AM	Column not found
16	≡	!	8:24:31.366 AM	Column not found
17	≡	!	8:24:31.358 AM	Column not found
18	≡	!	8:24:31.337 AM	Column not found
19	≡	!	8:24:31.329 AM	Column not found
20	≡	!	8:24:31.318 AM	Column not found
21	≡	!	8:24:31.311 AM	Column not found
22	≡	!	8:24:31.301 AM	Column not found
23	≡	!	8:24:31.294 AM	Column not found
24	≡	!	8:24:31.287 AM	Column not found
25	≡	!	8:24:31.273 AM	Column not found
26	≡	!	8:24:31.260 AM	Column not found
27	≡	!	8:24:31.244 AM	Column not found
28	≡	!	8:24:31.236 AM	Column not found
29	≡	!	8:24:31.228 AM	Column not found
30	≡	!	8:24:31.216 AM	Column not found
31	≡	!	8:24:31.203 AM	Column not found
32	≡	!	8:24:31.192 AM	Column not found


JOt automation
5/9/2023
9:54:35 AM
HOME
SERVICE
MES
VISION
ALARMS
DATA
ABOUT
Mode: Auto
AUTO
English
_SystemGuest
y-axis
x-axis
Start: First
End: Latest
Machine state: Error

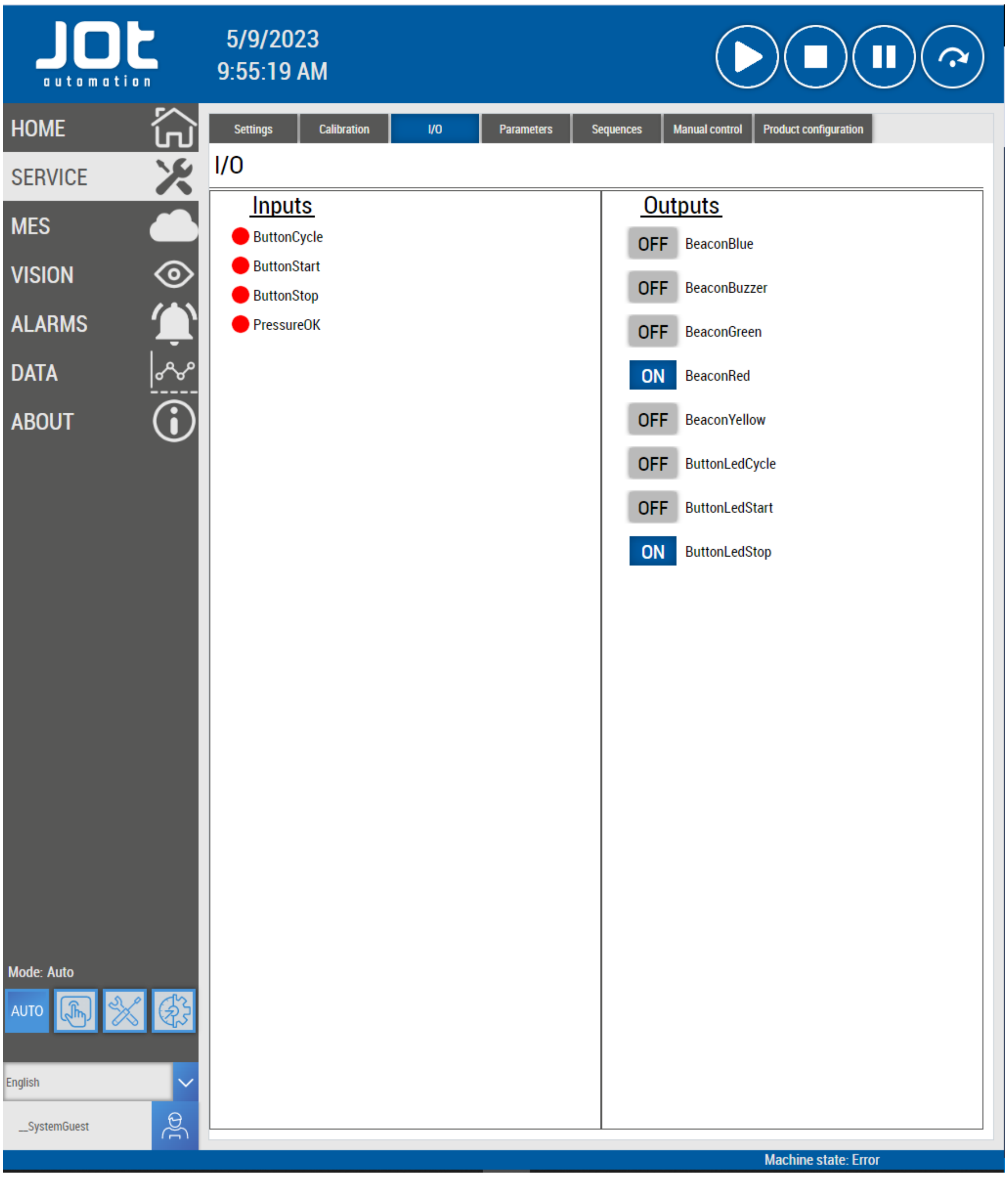


PLC

PLC project version	0.0.0
AMS Net ID	172.23.228.73.1.1
PLC HW Serial number	0000

HMI

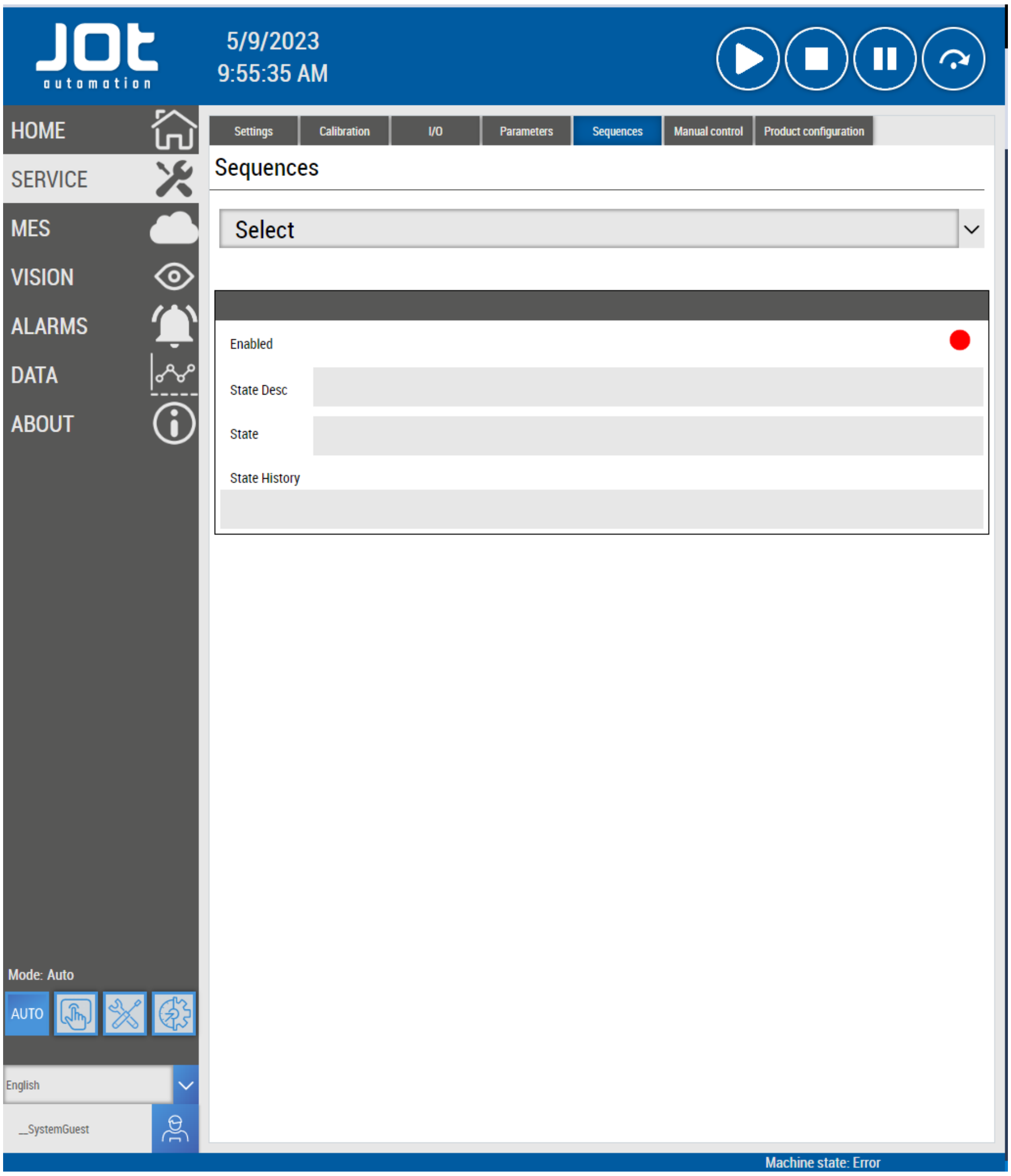
Software version	1.0.0.0



I/O

Inputs	
🟥	ButtonCycle
🟥	ButtonStart
🟥	ButtonStop
🟥	PressureOK

Outputs	
OFF	BeaconBlue
OFF	BeaconBuzzer
OFF	BeaconGreen
ON	BeaconRed
OFF	BeaconYellow
OFF	ButtonLedCycle
OFF	ButtonLedStart
ON	ButtonLedStop


5/9/2023
9:55:35 AM

Sequences

Select

Enabled ●

State Desc

State

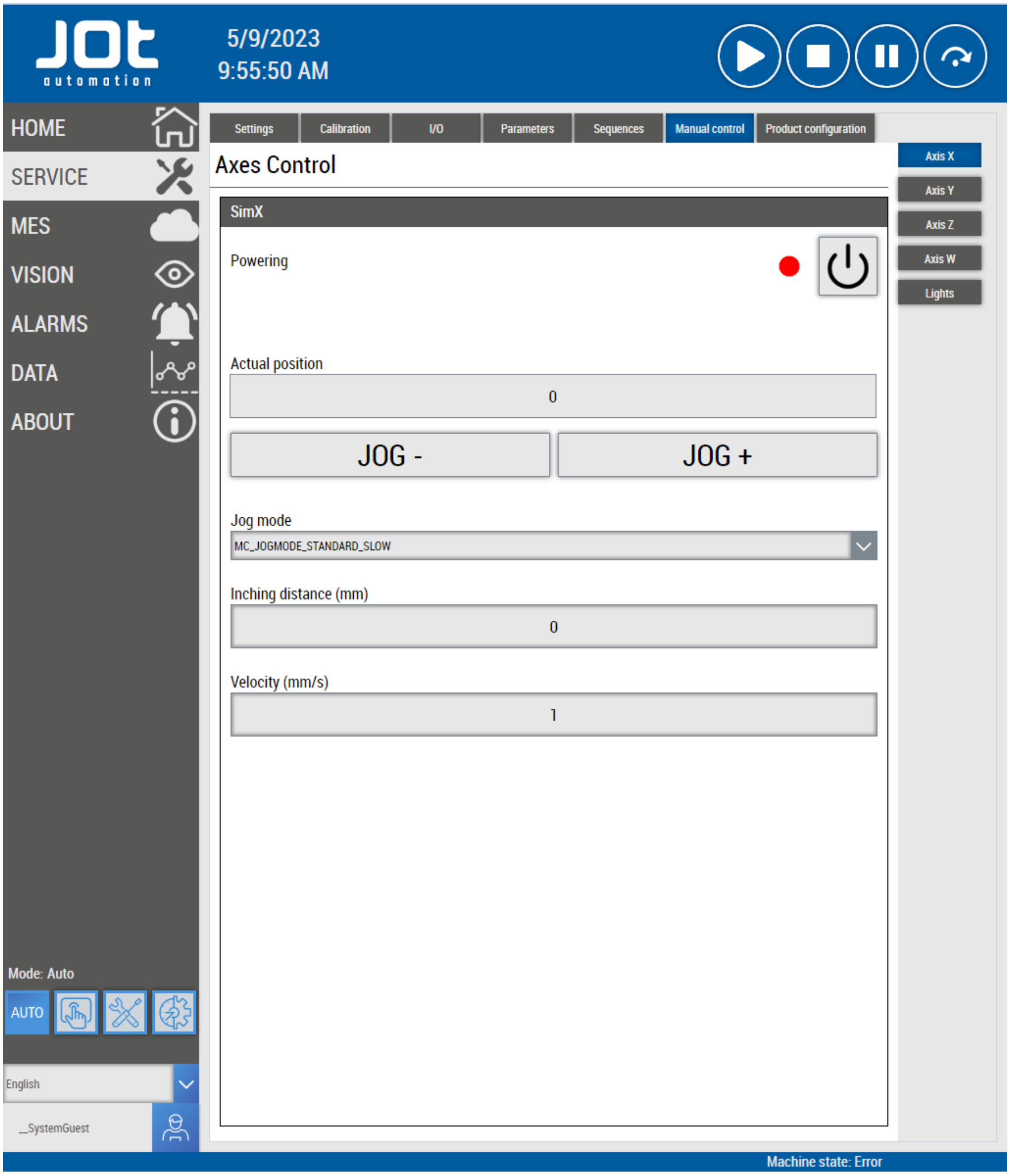
State History

Mode: Auto

English

_SystemGuest

Machine state: Error



Axes Control

SimX

Powering

Actual position: 0

Jog mode: MC_JOGMODE_STANDARD_SLOW

Inching distance (mm): 0

Velocity (mm/s): 1

Mode: Auto

Machine state: Error