Expertise
and insight
for the future

Markus Laitinen

Development of Virtual Test Environment for ABB Azipod Interface Unit

Metropolia University of Applied Sciences

Bachelor of Engineering

Electrical and Automation Engineering

Bachelor's Thesis

01 April 2020

Author	Markus Laitinen
Title	Development of Virtual Test Environment for ABB Azipod Interface Unit
Number of Pages	31 pages + 2 appendices
Date	1 April 2020
Degree	Bachelor of Engineering
Degree Programme	Electrical and Automation Engineering
Professional Major	Automation Engineering
Instructors	Antti Paukkio, Project Manager Timo Kasurinen, Senior Lecturer

This study was done for ABB Oy Marine and Ports, which develops electrification and automation solutions for the marine industry. The main objective of this thesis work was to develop a virtual testing simulator and an acceptance test (IAT) for internal testing purposes of the ABB Azipod X AIU (Azipod Interface Unit) product during the production phase. The ABB AC800M PLC is responsible for the internal logic operations of the Azipod X AIU. Using WinMOD software as the platform for the simulator environment was one of the key requirements of the study.

The WinMOD virtualization environment was used to develop the simulator and to perform the acceptance test for the PLC. Communication between the simulator and the programmable logic was performed using the OPC DA protocol. The testing program was based on the FAT/HAT testing procedures already in use, the most important sections of which were translated into WinMOD Scripts. These scripts could be executed in real time in the WinMOD simulator both manually and automatically.

As a result of the thesis work, a simulator platform was created for testing the AIU X product. This platform includes a WinMOD simulator as well as an internal acceptance test, both of which are easy to modify and expand. The thesis work proved the feasibility of WinMOD-based virtual testing concept, which can be further developed to meet different testing needs and procedures.

Keywords	WinMOD, HIL, PLC, Azipod, simulation, test

Tekijä Otsikko Sivumäärä Aika	Markus Laitinen Virtuaalisen testausympäristön kehitys ABB Azipod rajapinta yksikölle 31 sivua + 2 liitettä 01.4.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	sähkö- ja automaatiotekniikka
Ammatillinen pääaine	automaatiotekniikka
Ohjaajat	projektipäälikkö Antti Paukkio lehtori Timo Kasurinen

Insinöörityö tehtiin ABB Oy Marine and Ports -yksikölle, joka kehittää sähköistys- ja automaatioratkaisuja meriteollisuudelle. Työn tavoitteena oli kehittää virtuaalinen testaus simulaattori sekä hyväksyttämistesti (IAT) käytettäväksi yrityksen sisäisesti ABB Azipod X AIU -tuotteen testauksessa tuotannon aikana. Azipod X AIU toiminnasta vastaa ABB AC800M PLC. WinMOD-ohjelmiston käyttö simulaattorin pohjana oli yksi työn vaatimuksista.

Työssä käytettiin WinMOD-virtualisointiympäristöä simulaattorin rakentamiseen sekä hyväksyttämistestin suorittamiseen. Simulaattorin ja ohjelmoitavan logiikan välinen kommunikointi toteutettiin OPC DA -protokollaa käyttäen. Simulaattorilla suoritettava testausohjelma pohjautui jo käytössä oleviin FAT/HAT-testausmenetelmiin, joista tärkeimmät kohdat käännettiin WinMOD Script -komentosarjoiksi. Näitä komentosarjoja pystyttiin suorittamaan reaaliajassa WinMOD-simulaattorissa sekä manuaalisesti että automaattisesti.

Insinöörityön tuloksena syntyi simulaattorikokonaisuus AIU X -tuotteen testausta varten. Tämä kokonaisuus pitää sisällään WinMOD-simulaattorin sekä hyväksyttämistestin, jotka molemmat ovat helposti muokattavissa sekä laajennettavissa. Työ osoitti WinMOD-pohjaisen virtuaalisen testauksen toteuttamiskelpoiseksi. Työn pohjalta syntynyttä menetelmää voidaan jatkokehittää vastamaan erilaisia testaustarpeita.

Avainsanat	WinMOD, HIL, PLC, Azipod, simulaatio, testaus

Contents

List of Abbreviations

AIU	Azipod Interface Unit
Azipod	ABB Groups electric podded azimuth thruster
FAT	Factory Acceptance Test
HAT	Harbor Acceptance Test
HIL	Hardware-in-the-loop simulation
IAT	Internal Acceptance Test
IO	Input and Output
MAS	Machine Automation System
MCC	Motor Control Center
OPC	Open Platform Communications
OPC DA	OPC Data Access
PCS	Propulsion Control System
PLC	Programmable Logic Controller
VC	Virtual Commissioning
VDR	Voyage data recorder

1 Introduction

In this thesis work, a virtual test environment and internal acceptance testing procedure was developed for AIU (Azipod Interface Unit). This thesis work was done for ABB Oy Marine & Ports Finland.

Currently ABB Oy Marine & Ports Azipod AIU product is not officially tested during the engineering phase, which gives the opportunity to develop the process cycle and quality control. Internal AIU testing with current method is done manually and requires the AIU cabinet to be fully installed in the ship's engine room in order to test the control software. This adds to project quality costs due to engineering being done during the ships commissioning phase.

With help of the HIL simulator and testing programs, AIU testing can be executed at an earlier stage and faster than the current testing procedure is capable. The AIU testing program can be executed manually or automatically via the simulator before the final installation of the AIU control unit on board, thus saving time and resources, which ultimately will lead to decreased total cost associated to the AIU product. Using the simulator as a testing platform enables standardized testing practice and expedites products lead time.

1.1 Research Objectives

The objective of this thesis work was to develop a HIL-simulator to execute internal testing procedures for the AIU. Simulator must be able to simulate AIU auxiliary devices, Azipod internal operations and signal handling with capability to execute acceptance tests for AIU under office conditions. The purpose of the IAT testing is to verify the functional requirements set for the system are met and to ensure that the system functions as expected in case of failure. The IAT should include relevant points of the FAT/HAT tests and the test should be configurable with option to be expanded for other Azipod AIU products. The emphasis in the testing program will be on AIU auxiliaries and related functions.

Windows based WinMOD – program was chosen as simulator platform because of its traits being light to operate and promising specification. WinMOD is new tool in the

company, meaning that this thesis work is one of the first pilot projects where it is being utilized. One task of the thesis work was evaluate the suitability of the WinMOD as a standard testing tool and to map its scalability and adaptability to other Azipod products.

2 Use of Hardware-in-the-loop Simulation, HIL

Hardware-in-the-loop (HIL or HWIL) is a real-time simulation technique where real signals from a controller are connected to a virtual real-time implementation of physical components. Controller interacts with the simulator via chosen fieldbus and the simulation creates environment to test embedded systems software. This type of testing technique is widely utilized in the development and testing of complex control and monitoring systems in various industries. HIL simulation can be used to validate control procedures and behavior of target control system without physical process hardware as the simulation shows how the controller responds in real time to realistic virtual environment. In this virtual environment it is possible to model all the complex scenarios and related dynamic systems to validate the plant model and behavior of the controller. Virtual environment can simulate physical system in real-life situations or in extreme environmental conditions to test how controller responds to unusual events or abnormal process scenarios. This allows testing that could normally damage target system or otherwise jeopardize industrial safety without exposing hardware any real risks to the hardware. In a competent HIL testing the control and monitoring systems are unable to distinguish difference between the real process and the simulated one. [1;2;3.]

Typical applications for HIL simulation include:

1. Prototyping of machinery and automation systems

Prototyping systems with HIL simulation can be more economical and more practical than conventional prototyping as there are no need to invest in real hardware at the beginning of the product development. Simulator can be also substantially faster to implement, modify and replace than the actual hardware. In other cases, it is possible to utilize the simulator as a substitute of unavailable parts of the real system. [4;5.]

2. Software development

Simulation can be performed early in the development cycle to identify weak points and to develop solutions for found problems. Initial designs can be improved, and alternative plant models can be explored. Design changes and modifications to the simulator are likely less costly to implement than real prototype and there is more time to redesign process in early stage which can cut engineering costs down in later stages of production. [3;6.]

3. Validation of the software through simulation

Validation of the control software can be achieved without exposing hardware or equipment to danger while testing control algorithms and the validations can be done without expensive downtime of the physical system [4]. Testing and validating can be performed without first-hand knowledge of the internal structures of the control system as in HIL testing the control system is viewed as a black box which means that no source code is exposed during the testing. As a result of this method HIL testing is relevant in every level of software testing. Test cases for the target system are developed based on functional descriptions, user manuals, class rules and regulations, and industry standards. These testing cases can cover fail-safe testing, auxiliary systems testing, parameter sweep testing, system proofing etc. [7.]

4. Operational and maintenance training with simulated components

Well-designed graphical simulator can be used as training tool for operators and engineers in a risk-free environment e,g. simulators for aerospace industries and critical systems such as a nuclear power plants. Some plant operations or systems can be complex and delicate which could make training with the real equipment impractical or could cause resources and business loss. With the training simulator the operators are familiarized with the plant operation without effecting the production flow. [5.]

2.1 Use of HIL Simulation

HIL simulation is utilized in the automotive, aerospace, defence and industrial automation to test control systems and products as the physics models and simulations can provide very reliable results between simulated and real-world phenomena. Reduced

development costs and increased functionalities are the main selling points for various industries. [2.]

The EPC Power Corp utilized HIL simulator to develop battery storage inverter control software due to increased demands of the testing procedures. Conventional testing in a testing laboratory has its limitations and testing larger systems by conventional means requires more space and power. HIL simulator was used to execute preliminary system testing and later to run PVT (Performance Verification Test) for inverter control software. Using simulator decreased testing times of the control software significantly, according to their studies they were able to perform software-hardware integration for 500-kW inverter in one day instead of six weeks using HIL simulation. [8.]

In 1998 the Sandia National Laboratories created WinMOD casting simulator for the FASTCAST consortium. Objective of the work was to create tool to support the decision-making, visualize the parts and cooling process, in order to identify problems in casting process, eliminate possible casting defects and to provide aid for optimal placement of process elements. Heuristic WinMOD simulation and visualization of relative solidification process were utilized in several casting applications resulting slightly lower material losses in the metal casting processes. [9.]

In [10] ABB's Marine industries has built Integrated Marine Systems (IMS) laboratory utilizing HIL simulation technology. This IMS platform contains the key components of ABB's marine technologies, dynamic characteristics of the electrical power systems, low level power electronics controllers and protection devices together with real logic-containing controllers i.e., all the critical equipment and connections as in the actual vessel. Using IMS Test platform enables ABB to perform testing and to improve overall system efficiency and quality of these marine systems in a controlled environment before actual delivery on board. System can be tested in normal and fault situations on variety of scenarios and sea conditions. IMS test platform is utilized in the development of new technologies as well.

2.2 Summary

HIL Simulation is industry standard for testing and validating control system software due benefits in costs and practicality. Advantages of using simulator for testing purposes:

- Enhances Quality and Safety:
    1. Tests can be performed without risk to people, equipment or environment
    2. Increased test coverages due no real-world limitations
    3. Standardizes test procedures
    4. Eliminates critical errors affecting to real processes.
- Saves Time and Money:
    1. Faster to build than fully physical prototype
    2. Shortens testing times
    3. Reduces the troubleshooting process
    4. Easier software modifications
    5. Low cost implementation.
- Automates Testing:
    1. Automatic test procedures
    2. Automated environments
    3. Removes human factors.

HIL simulation shows advantages of using real-time models all kinds of use cases, from simple signal simulations to complex scenarios. Low cost implementation of sophisticated virtual models and comprehensive testing are revolutionizing the future of production of embedded systems. [5; 11.]

3 WinMOD

WinMOD is a Microsoft Windows based simulation platform developed by Mewes & Partner GmbH. It is capable of virtualizing single components in addition to complex automation systems in real time including wide range of communication protocols needed to connect to the real control systems or other engineering tools. WinMOD is a visual programming environment which uses functional block diagram and simple scripting language. WinMOD real-time simulation platform enables transparent monitoring and immediate interaction between operator, simulation and automation system. Virtualized systems can be used to completely replace their real-world counterparts, or they can be used in conjunction with existing automation systems to form fully operational automation process (Figure 1). These hardware-in-the-loop simulation capabilities enables the WinMOD to be utilized in the development, testing and the virtual commissioning of the automation systems. In this thesis WinMOD was used to create virtualized test

environment for AIU X controller and to execute testing procedure for it. WinMOD was chosen as simulation platform because of portability and promising potential as it allows engineering, real-time simulation and monitoring done in one program. [12; 13.]

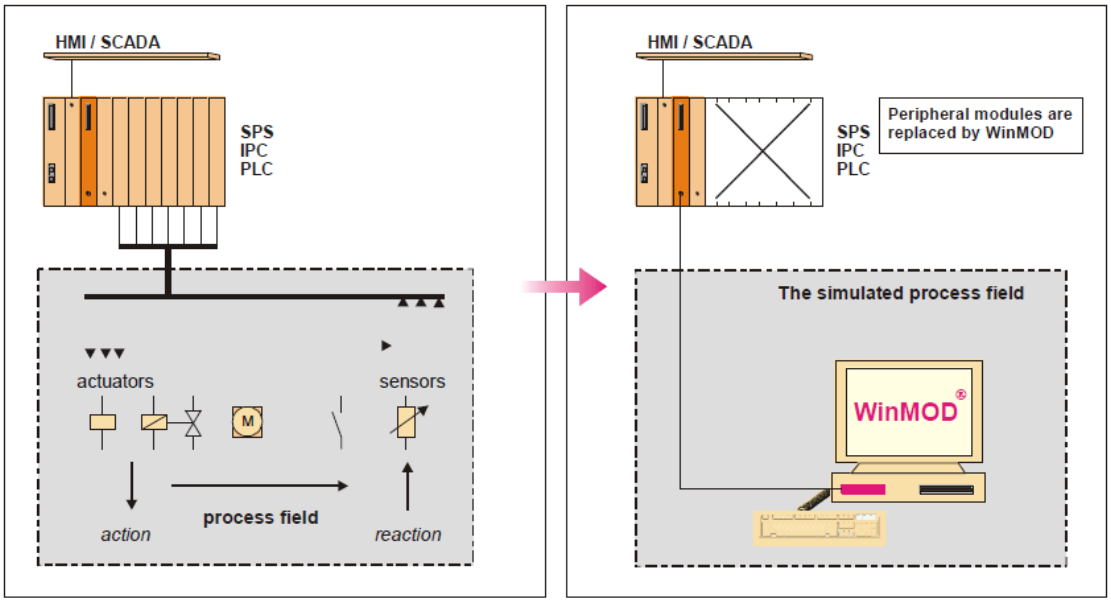


Figure 1. WinMOD simulates real process field [18].

Relevant features of the WinMOD System Software for this thesis work:

- Comprehensive simulation elements from simple logic to advanced formulas
- Simple macro creation
- Ability to model virtual machines and plants
- OPC connectivity (Figure 2)
- High-performance online visualization, online forcing and parameterization
- Ability to run simulation parallel to the real system in real time.

For this thesis ability to model AIU auxiliary devices and connect to the AIU PLC were the most important points. Macro creating was utilized in testing phase along with the visualization abilities. [14;15.]

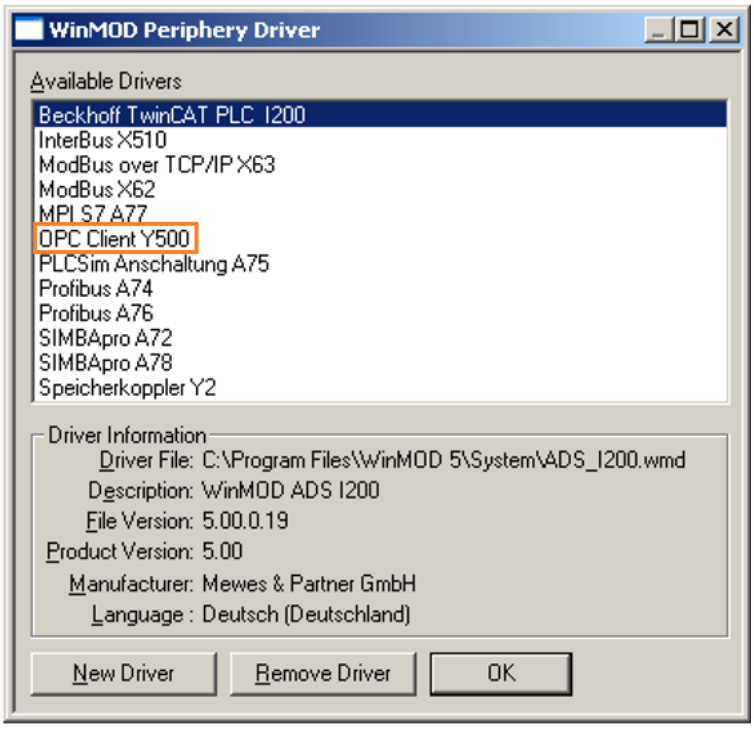


Figure 2. WinMOD Connectivity Drivers. OPC Client Y500 was used in this thesis work.

Recommended system requirements for WinMOD:

- Microsoft Windows from Windows XP 32-bit (from SP3) to Windows 10 64-bit
- standard PC with Intel Core i7 of the current generation from 3.5GHz base clock
- at least 8GB of RAM
- at least 2GB of hard disk space
- One free USB port for licence dongle.

Laptop used in this thesis work meets all other requirements except the recommended processor, Intel Core i7. Performance was relevant question because system running WinMOD had Intel i5 with base clock of 1.9 GHz. According to [16] it was about 57% slower than the recommended i7 processor. However, during the implementation phase, it was found out that performance of the system was not an issue. [17.]

Interactive graphical interface with element library (Figure 3) is used to create virtual machines and systems by utilizing WinMOD simulation elements. These elements can be used to build virtualized binary and analog signals and their behaviour. Simulating

drives, measuring systems, signal devices or other process signals can be achieved by linking these simulation elements together. Graphical interface allows graphical design, observation and forcing of the signals and elements in real time. Depth and complexity of simulation is determined by requirements of the target system.

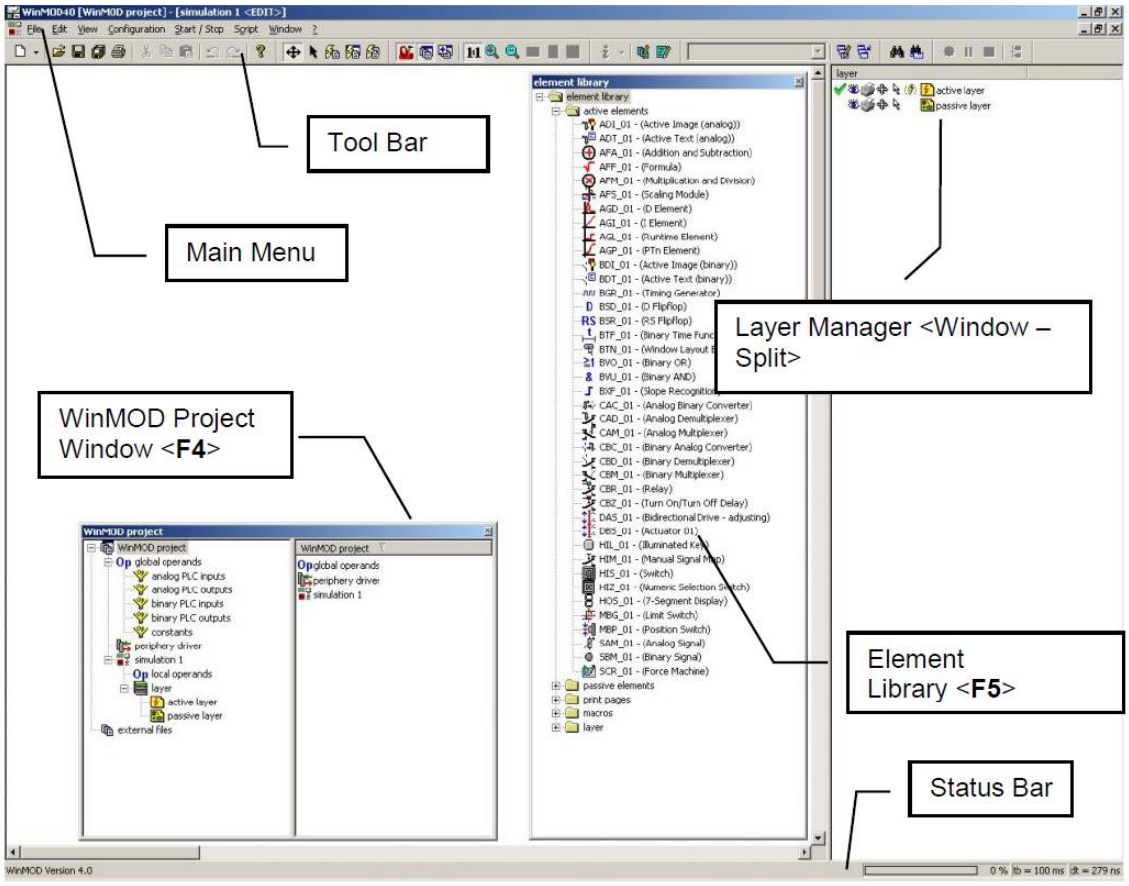


Figure 3. WinMOD main window and element library [18].

3.1 WinMOD Script Syntax

WinMOD Scripts are a domain-specific scripting language which allows time-defined control of WinMOD elements. These script files can be identified from header line in the text file. Script Recording function in WinMOD allows recording operating processes on WinMOD elements as scripts sequentially. WinMOD scripts can be also created and edited in any text editor using the syntax presented in Listing 1. Force Machine simulation element can be used to execute these scripts in Run mode. [18, 89-93.]

```
Command name Parameter1[ Parameter2[... Parameter n]]
```

Listing 1. WinMOD Script Syntax.

Each script command must be in a separate line as the script is interpreted line by line. Script files can contain comments and following commands: WAIT, CMD, INCLUDE, TEXT. Example of a basic WinMOD Script is presented in Listing 2. [18, 93-95.]

```
#WinMOD-Script V1.0        ;Header

; Initialization, activate operand forcing
cmd "SBM_01 1" F=1 V=0     ;activate forcing and set signal value 0
cmd "SAM_01 1" F=1 V=0.0   ;activate forcing and set signal value 0.0
cmd "SDM_01 1" F=1 V=0     ;activate forcing and set signal value 0

; Setting value to operand
wait 1000                  ;wait 1000ms
cmd "SBM_01 1" V=1         ;set signal value 1        ;Binary Signal
cmd "SAM_01 1" V=12.25     ;set signal value 12.25    ;Analog Signal
cmd "SDM_01 1" V=10        ;set signal value 10       ;Digital Signal

; Deactivate operand forcing
wait 5000                  ;wait 5000ms
cmd "SBM_01 0" F=0         ;deactivate forcing
cmd "SBM_01 1" F=0         ;deactivate forcing
cmd "SAM_01 1" F=0         ;deactivate forcing
TEXT " NORMAL STATE"       ;Text command
```

Listing 2. Example of WinMOD script.

3.2 Signals and Operands in WinMOD

WinMOD can handle three types of signals: binary signals, digital signals and analog signals. AIU auxiliary signals were mainly analog 4–20 mA and digital 24V signals. Most of the analog signals came from PT-100 temperature sensors. In order to handle signals in WinMOD, they need to be transformed into the correct format. In Figure 4 conversion table for the operands can be observed. In this case AI and AO operand types were used for analog signals and respectively BI/BO for binary signals and DI/DO for digital signal.

Operand Types

When importing operand lists into the range of global or local operands, you must indicate the operand type as type declaration.

Operand Type Declaration

The following symbols are defined for the type declaration:

BI	–	AS binary input
BO	–	AS binary output
AI	–	AS analog input
AO	–	AS analog output
DI	–	AS digital input
DO	–	AS digital output
BC	–	binary constant
AC	–	analog constant
DC	–	digital constant
BM	–	binary process operand
AM	–	analog process operand
DM	–	digital process operand

Figure 4. WinMOD Conversion table for automation system signals [18].

4 Presentation of the AIU X System

AIU X is part of Azipod X product delivery and is responsible for data handling of the Azipod modules and control of the Azipod auxiliary devices. In Figure 5 basic arrangement for Azipod X system can be seen. Typical Azipod X delivery consists of two main modules, Steering and Propulsion, along with 11-15 auxiliary systems depending the intended use of the ship and operating environment, (open water conditions or icy conditions). In this chapter AIU X system and its functions are presented. [19.]

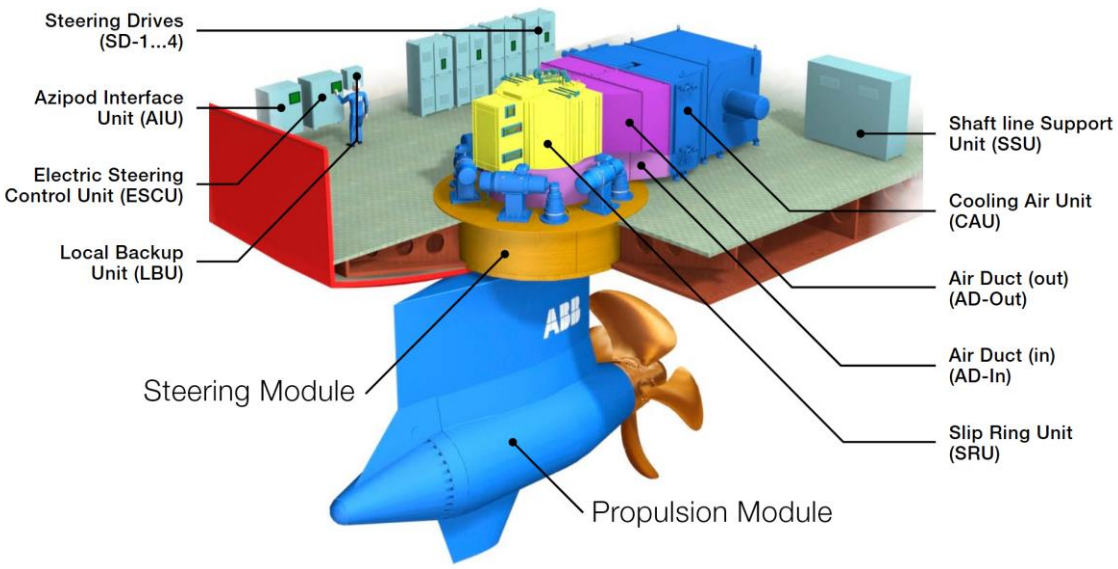


Figure 5. Azipod X modules and auxiliaries [19].

4.1 AIU X System Delivery

Azipod Interface Unit (AIU) is essential system in Azipod X delivery. AIU system itself is a 200 kg cabinet (Figure 6) which houses two ABB AC800M PLCs for redundancy, HMI panel for monitoring and control of Azipod subsystems, ethernet switches, circuit breakers, power supply modules and data transmission modules. Total of two AIU units are installed for each Azipod propulsion units in the Azipod room. These AIUs has different roles and functions, where other is responsible for Azipod steering control and the other is responsible for Azipod interface and auxiliary control [20]. In this thesis focus of the work is the controller responsible for control of auxiliary devices.

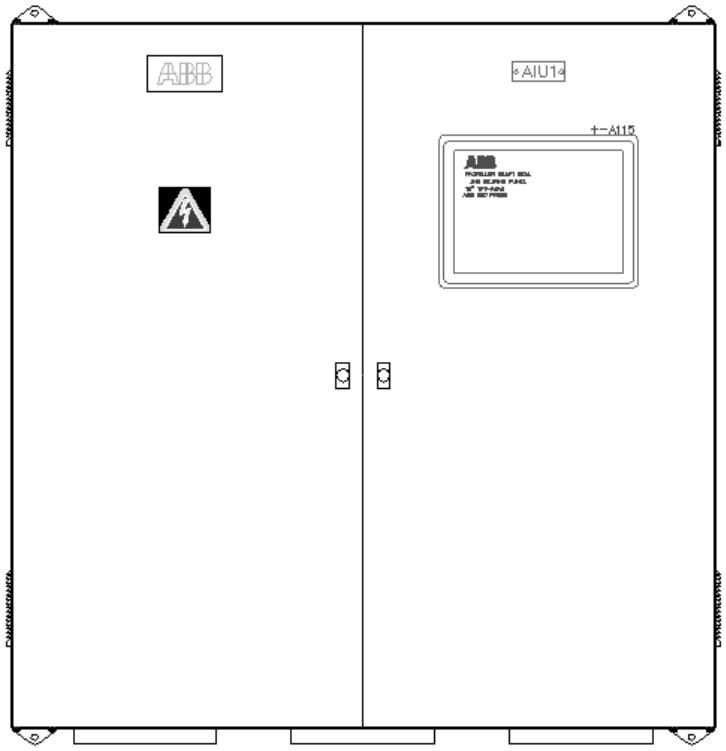


Figure 6. AIU X cabinet [21].

4.2 Functions of AIU X Programmable Logic

AIU in the propulsion control network controls the auxiliary systems and is the interface between Azipod systems and the ship's automation system. AIU PLC is responsible for:

- Handling data transmissions between Azipod modules
- Data delivery to PCS, MAS and VDR

- Interface between external systems (bridge control and ship automation)
- Control of propulsion auxiliaries (pumps, fans, space heater, greasing unit, valves.)
- Control of cooling air subsystem
- Group monitoring and alarms generation from Azipod propulsion module and its auxiliaries

Control software holds functions for interlocks, start/stop control, auxiliary control, duty/standby control, protection protocols, reaction to failures and blackout recovery. Communication between AIU and ships automation system is routed through Modbus RTU protocol or via TCP data communication. Communication is mainly handled by the AIU OPC server. In Figure 7 layout of the interaction between Azipod modules and ship's machinery is shown. Auxiliary device control is handled by the AIU and executed by the ship's machinery and automation, more specifically by the motor control center (MCC). Basic principle is that the AIU sends the control command, CMD 0 or CMD 1, to the MCC which then starts or stops the auxiliary device. Some of these control commands can be send using the HMI, but mainly the control is handled by the control software automatically. In this thesis WinMOD is used to simulate behavior of the MCC in order to test the AIU control software. [21.]

HMI in the AIU cabinet, Azipod local panel (ALP) is used to monitor and control the AIU auxiliaries. ALP functions include:

- Controls for propeller shaft water seal system (operational mode, filling, draining and parameters)
- Indication of seal tank level
- Indication of drainage levels
- Controls for the system (filling and draining)
- Indication of the bearing oil sumps levels
- Indication of the bearings and thrust pads temperatures
- Indication of the shaft seal and bearing oil systems
- Indication of the shaft line support unit's (SSU) operating values
- Control of the shaft line support unit parameters
- Indication of the propulsion motor's temperatures
- Indication of the states of the shaft accessories

- Indication and control of the cooling system parameters
- Control of several service/system maintenance functions
- Alarm listing of the Azipod sensors. [21.]

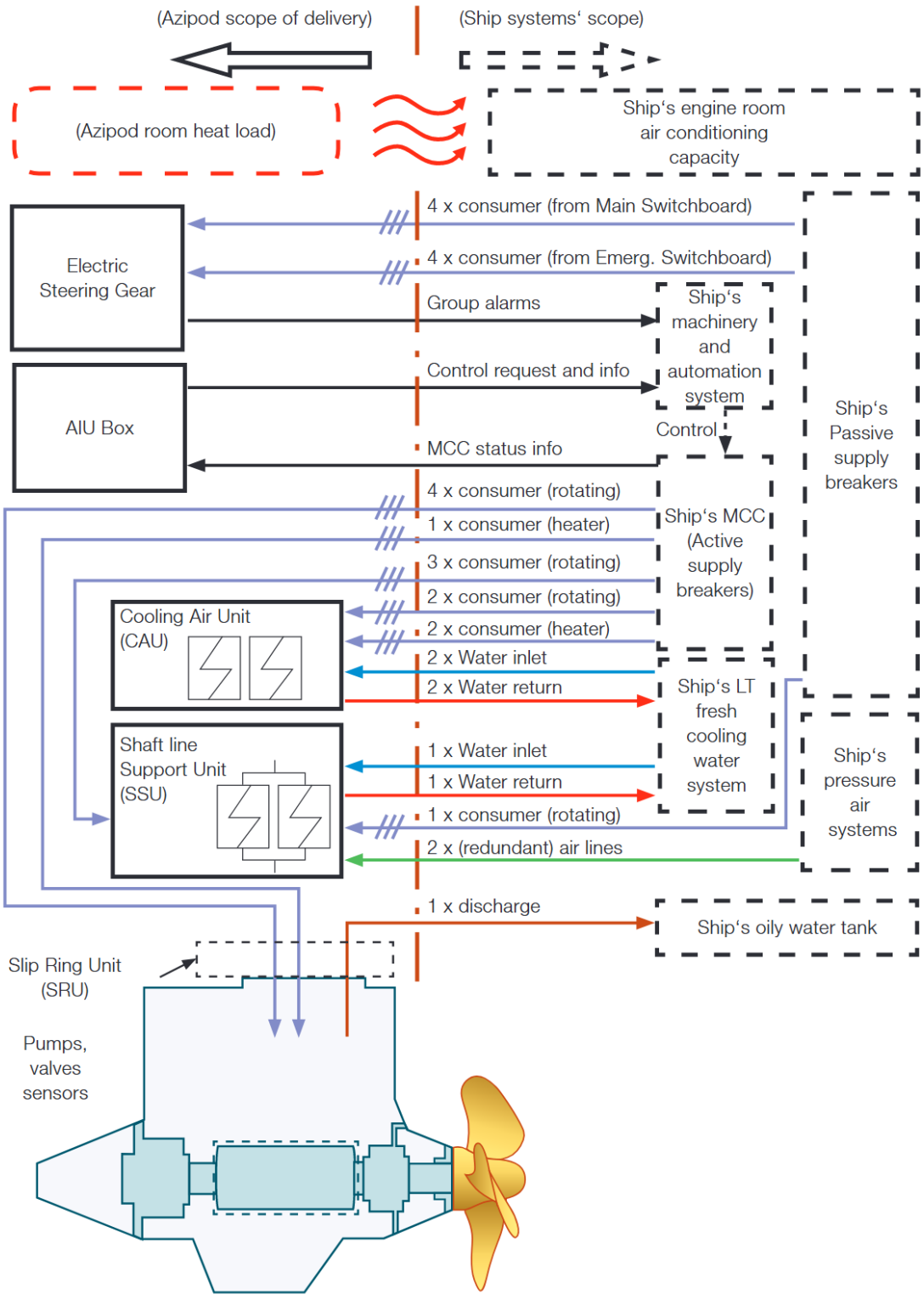


Figure 7. Layout of the interface between Azipod modules and ship's automation systems [22].

Azipod AIU PLC contains the software for controlling the Azipod auxiliary equipment and 800xA libraries and graphics. The module also generates limitations for the PCU (Propulsion Control Unit) system according to the equipment status. The software consists

of a set of different system libraries of which the main program is built. Some of the modules controlled by the AIU are listed below.

- Pressurization system and air feeding
- Leakage Detection System
- Cooling unit
- Drainage water system
- Shaft Line Brake and Locking
- Slewing Bearing Oil Circulation System
- Slip ring unit fans
- Drainage pump
- Temperature and vibration sensors
- Bearing System magnetic coupling pumps
- Azipod Pit lights. [21.]

These modules includes motors for pumps and valves, sensors, heaters and fans. Majority of devices tested in this thesis were motors and different sensors. Mainly the sensors and signals tested were PT-100 temperature sensors from different auxiliary systems.

5 Building the AIU Simulation Platform

In chapter 4 hardware, operation principles and environment of the AIU X was presented. This chapter deals with definition, design and implementation of the AIU X Simulator. The initiation phase defines what kind of functionalities and features would be required from the simulator and what its intended use would be. In the planning phase simulation software were designed according to the definition to provide required functionalities for the simulator. The implementation phase describes how the simulator was built on the WinMOD environment and the testing procedures are discussed at the end of the chapter.

5.1 Initiation Phase

The objective of this thesis work was to create testing tool to shorten the testing time of the AIU X product and to improve quality by unifying the testing procedures. This testing

tool, WinMOD simulator, was designed to be a separate system which connects to the AIU controller via OPC server. This software-based simulator enables the testing of controller IO without access to the controller's source code or use of the developer software. The WinMOD software was chosen as simulator platform, because of its promising features as a testing tool. WinMOD can be installed on a regular laptop, which is beneficial in many phases in the production cycle of the AIU X product. In this thesis work WinMOD is used to execute acceptance tests for ABB's internal use.

Benefits of using simulator in general was discussed in chapter 2. One the main motivation in this case is easing and shortening time spent on testing the AIU X product. Basic Azipod delivery consists of two main modules, propulsion and steering, and eleven to fifteen auxiliary modules which are separately build and separately installed onboard by the shipyard [20]. AIU unit is one of these auxiliary modules. Being separately installed systems, AIU cannot be fully tested before the environment is fully built and everything is connected. Simulator, WinMOD based simulator, virtualizes the this fully built environment and devices needed to execute acceptance tests for AIU controller before final installation of the AIU unit.

Initiation phase started by defining desired functionalities and features for the simulator along with the example use cases. The purpose of the simulator and the project objective / problem statement were quite clear so the scope of the work could be determined with fair accuracy. Main objective was to create testing platform for AIU X product using WinMOD software. With this WinMOD simulator virtual AIU X auxiliary devices could be connected to the AIU X controller and the behaviour of the control software could be tested.

The simulator was supposed to be light, portable and easily customizable. Based on previous tests done by ABB, the WinMOD software was chosen as the platform for the AIU simulator. WinMOD uses relatively low amount of system resources of the computer where it is running, so it can be utilized effectively on a regular laptop with Windows OS. Use of WinMOD ensured portability and easily configurable test environment for wide range of testing. In addition to the implementation of the simulator, the functionality and applicability of the WinMOD software as a testing tool would also be studied.

5.2 Planning Phase

Planning started with developing solution for stated problem further in finer details. Simulator had to be simple enough to use without WinMOD training and design of the user interface had to be clear. Test programs had to be easy to select and modify if needed. Few functionalities were added to the plan, most important being automatic testing capability.

The following key features were planned for the simulator:

- Ability to test real ABB AC800 PLC and software PLC of same variant
- Ability to simulate auxiliary devices and systems of the AIU module
- Ability to test all relevant IO signals
- Ability to test all functions of the AIU X control software
- Ability to execute tests automatically

In order to test AIU control software AIU auxiliary systems and devices had to be virtualized in WinMOD along with corresponding functionalities and IO signals. These auxiliary systems included motor pumps, valves, heaters and wide range of different sensors. The IO signals were mainly analog and digital boolean values.

The functions of the virtualized components had to be equivalent to those of the physical counterparts and had to be equivalent to the physical devices from the point of view of the AIU control software to eliminate unnecessary alerts caused by misbehaving virtual devices during the testing and to ensure quality of the testing procedure.

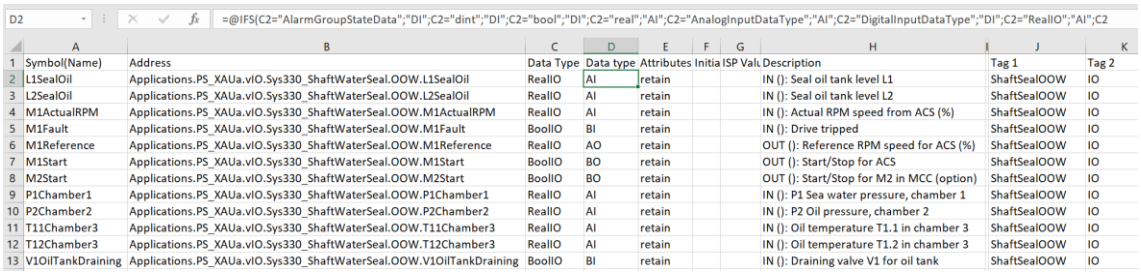
WinMOD is used to simulate IO signals and run the test procedure for the AIU. Laptop with WinMOD simulator connects to the AIU PLC using ethernet connection and data exchange is done via OPC server as an OPC client. WinMOD Configurations Y500 periphery driver enables connection between WinMOD simulation and real automation system for real-time simulation and allows WinMOD to read/write from/to OPC items if corresponding rights has been granted by OPC server. AIU AC800 PLC uses OPC DA (Data Access) protocol in OPC server communications [23]. Base for the simulator was Intel i5 powered laptop with WinMOD System Software V7.2

5.3 Implementation Phase

Work began by conducting a background study for AIU X auxiliary devices and 800xA control software. Functional descriptions of the individual modules along with the 800xA AIU X Library was main source of information for the research. Functional descriptions contained module's intended capabilities, functionalities, alarms, devices, signal lists, appearances, and interactions with control system and/or users. The code in 800xA AIU X library specified the information, most significant refinement being amount and data type of IO along with the functionalities. These sources served as guideline and reference point as the implantation of the simulator progressed.

Next step was to study how WinMOD operates and handles the data. Preconstructed examples from WinMOD OCA libraries along with WinMOD manual were great pool of information that helped to smooth the learning curve of the software. Towards the end, the OCA library turned out to be a better source of information because the WinMOD manual was somewhat lacking on the technical side due to errors in translation from German into English.

All essential IO were imported from the 800xA AIU X library into Excel file where they were transformed into a form and operand type that WinMOD requires using Excel formula (Figure 8). In the process they were provided with an identification tag to facilitate later processing in the WinMOD environment. WinMOD supports up to a five user defined tags three of which were utilized to sort and filter IO-signals in WinMOD global operand list. Imported 800xA IO signals contained following information: Name, Address (OPC), Data Type, Attributes, Initial Value, ISP Value and Description. Excel formula was created to automate data type conversion, because a manual conversion would have been too time consuming. Converted signal list was then imported to WinMOD as a comma-separated values (CSV) file.



D2 =@IFS(C2="AlarmGroupStateData";"DI";C2="dint";"DI";C2="bool";"DI";C2="real";"AI";C2="AnalogInputDataType";"AI";C2="DigitalInputDataType";"DI";C2="RealIO";"AI";C2

	A	B	C	D	E	F	G	H	I	J	K
1	Symbol(Name)	Address	Data Type	Data type	Attributes	Initia	ISP Val	Description		Tag 1	Tag 2
2	L1SealOil	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.L1SealOil	RealIO	AI	retain			IN (): Seal oil tank level L1		ShaftSealOOW	IO
3	L2SealOil	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.L2SealOil	RealIO	AI	retain			IN (): Seal oil tank level L2		ShaftSealOOW	IO
4	M1ActualRPM	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.M1ActualRPM	RealIO	AI	retain			IN (): Actual RPM speed from ACS (%)		ShaftSealOOW	IO
5	M1Fault	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.M1Fault	BoolIO	BI	retain			IN (): Drive tripped		ShaftSealOOW	IO
6	M1Reference	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.M1Reference	RealIO	AO	retain			OUT (): Reference RPM speed for ACS (%)		ShaftSealOOW	IO
7	M1Start	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.M1Start	BoolIO	BO	retain			OUT (): Start/Stop for ACS		ShaftSealOOW	IO
8	M2Start	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.M2Start	BoolIO	BO	retain			OUT (): Start/Stop for M2 in MCC (option)		ShaftSealOOW	IO
9	P1Chamber1	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.P1Chamber1	RealIO	AI	retain			IN (): P1 Sea water pressure, chamber 1		ShaftSealOOW	IO
10	P2Chamber2	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.P2Chamber2	RealIO	AI	retain			IN (): P2 Oil pressure, chamber 2		ShaftSealOOW	IO
11	T11Chamber3	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.T11Chamber3	RealIO	AI	retain			IN (): Oil temperature T1.1 in chamber 3		ShaftSealOOW	IO
12	T12Chamber3	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.T12Chamber3	RealIO	AI	retain			IN (): Oil temperature T1.2 in chamber 3		ShaftSealOOW	IO
13	V1OilTankDraining	Applications.PS_XAUa.vIO.Sys330_ShaftWaterSeal.OOW.V1OilTankDraining	BoolIO	BI	retain			IN (): Draining valve V1 for oil tank		ShaftSealOOW	IO

Figure 8. Excel I/O list and conversion sheet.

Core components and essential functionalities were modelled into the simulator software and analog signals were assigned their corresponding variables, i.e. temperature, pressure, flow. Some functionalities required interaction between devices (pumps, valves, sensors) and IO signals. One of these was oil tank simulation element in one subsystem where the level of the tank surface had to vary according to the operation of the pump and the valve. WinMOD macro named Analog Signal Simulation was created for this case. This macro consists of five inputs, two outputs and mathematical function which calculates changes in the analog signal. In this Oil Tank Simulation operand macro is used to simulate level of the oil in the tank in real time to verify control functions of pump motor and valve of the oil tank. WinMOD monitors alarms and command orders sent from the AIU to MCC, which is in this case simulated with WinMOD. Basic function principles of Azipod X were discussed in chapter 4. For simpler signal changes, one function block with simple math function was enough.

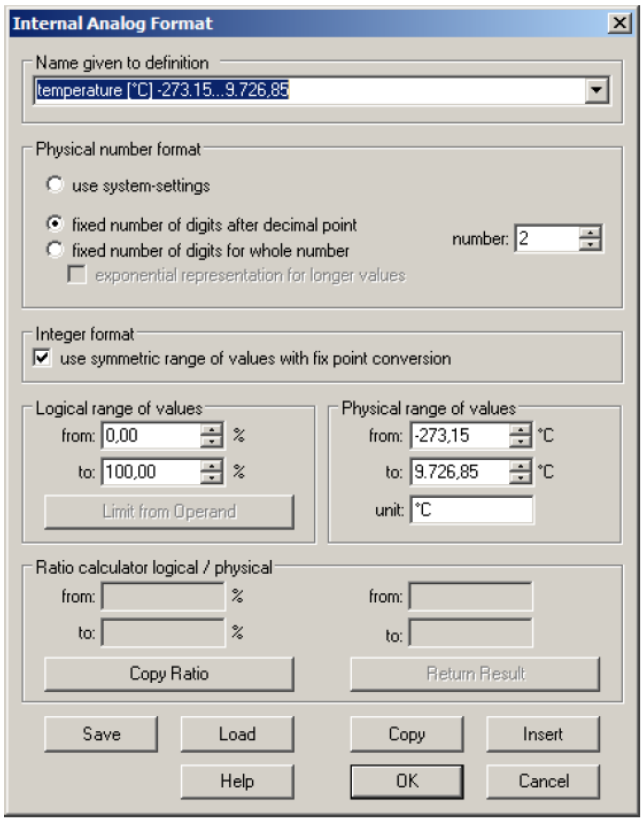


Figure 9. WinMOD Internal Analog Format -window.

Design of the user interface and user experience was kept simple and intuitive through implementation of the simulator. Main objective was to keep all relevant data visible at a glance and to keep layout of the simulation pages similar or the same as another. Being graphical interface, user experience needed to be practical, efficient and easy. In principle, only object user needs to operate is the Force Machine element, so it was made distinguishable large and placed at the same location in every testing page. Automatic testing and testing by steps are possible with these blocks (Figure 10).

Limit values for analog signals, alarm triggers and alarm signals were imported from 800xA AIU X library to the simulator in order to execute perform testing. To monitor triggered AIU alarms during testing, Alarm Detector macro was built. This macro consists of set/reset operands which are connected to AIU alarm signals. Like the other WinMOD macros, this can be copied and used in many different scenarios. Other options for monitoring test results with more advanced data logging capabilities were explored for this but weren't feasible due to additional licence requirements as WinMOD Logger is needed in order to export any data out of WinMOD software. For this thesis work Alarm Detector macro was competent enough.

5.4 Testing and Closing Phase

Executing testing scripts in WinMOD is done by utilizing SCR_01 – Force Machine (Figure 10) which allows the manipulation of WinMOD elements in the Run mode based on previously recorded or manually generated WinMOD script files.

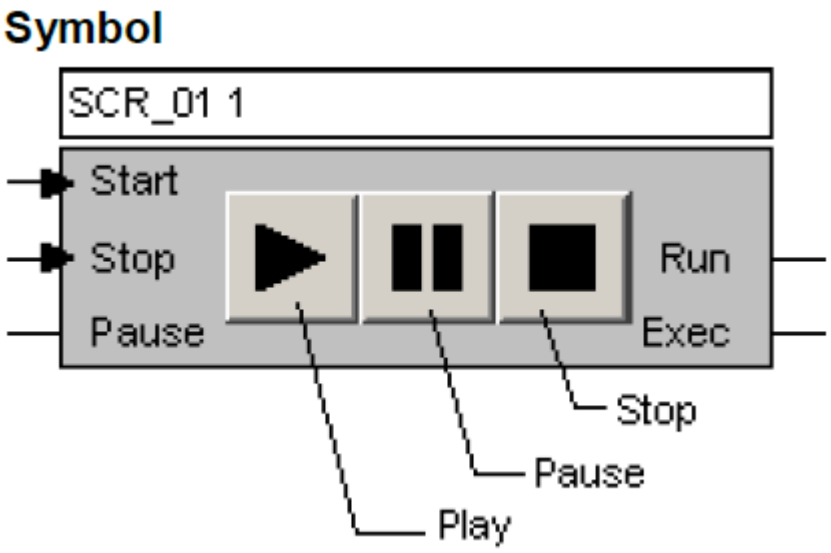


Figure 10. WinMOD Force Machine SCR_01 [18].

WinMOD script files can be executed in automatic mode or in the step-mode where script command WAIT serves as break point for the force machine. WinMOD script syntax is quite simple and scripts files can be edited in any text editor.

Few different test cases were selected from ABB PCU Software FAT (Factory Acceptance Test) and Dock Trial procedures and were converted into WinMOD script. These tests contained three typical test cases which could later be used to build larger testing script that covers the entire FAT. Similar test items can be found at SAT program and Internal AIU Testing programs.

Test included following items:

- Auxiliary Devices Duty/Standby Functionality
- Start interlocks: Oil level interlock simulations
- Power-limitations: Simulated temperatures

Auxiliary Devices Duty/Standby -test verifies duty/standby functionality of the auxiliary devices using HMI inputs given by operator. In this case it was specifically selected to study if WinMOD was able to act as virtual HMI (Human-Machine Interface) and virtual operator. These inputs were digital signals which WinMOD was able to simulate without difficulties. Other tests included digital and analog signals of which an example test procedure is shown in the chapter 6.4. WinMOD handled these two other tests without difficulties as well.

Simulator was able to run these FAT items and proof of concept for the WinMOD based testing simulator was verified. AIU can be tested with AIU simulator and FAT/SAT programs can be executed with the WinMOD. Regular laptop is powerful enough to run WinMOD and simulations in real-time as there were no performance issues detected during the testing. However, there were communication problems between the OPC server and the simulator during testing due to OPC server instability. OPC server kept crashing and terminating all connections after a few pull/push actions, which could be caused by incorrect settings or some other fault in the system. After investigating the problem more specifically by adjusting scan/update rates and other server/client settings, the most likely cause for crashing was some sort of memory leak in the OPC server. OPC server was running in the software PLC in the virtual (Windows 10) machine and the simulator software was running in the physical machine, which in its entirety were probably the root cause for problem. VMware Workstation 14 Player was used for virtualization. Communication issues between AIU OPC server and WinMOD needs to be addressed in the future development of the simulator if this is intended to be used as a testing tool.

6 Structure of the AIU Simulator

Simulator consists of two different page types, individual system pages and testing pages. Purpose of individual system pages is to present all available IO and functions of the system. Testing pages are used to construct test procedure and execute the test scripts. All the pages can be run individually or in group. For this thesis work all the testing pages were created in the same simulation file in order to simplify the process. In this chapter structure and layout of the simulator is presented. In this chapter structure of the simulator is presented.

6.1 Main Testing Pages

For this thesis three different test procedures (Figures 11-13) were modelled in the simulator using WinMOD script and force machine as were discussed in chapter 5. Basic principle of the pages is the same, all relevant data are shown in the page and the test procedures can be executed in STEP or AUTO mode. Variables for test procedures and individual simulation elements can be modified if needed. Test script is embedded in Macro block and it can be edited in the WinMOD using force machine editor or externally using any text editor.

In Figure 11 Drive-End Bearing test from Dock Trial Procedure – test document has been modelled in WinMOD. Force Machine for test script execution can be found in upper right corner and the AlarmDetector macro is located under them. Test IO and variables are located at the left side of the page. Test procedure verifies the functions of the drive-end (DE) bearing lubrication system by simulating oil temperatures, oil pressures, oil levels and behaviour of the oil pumps.

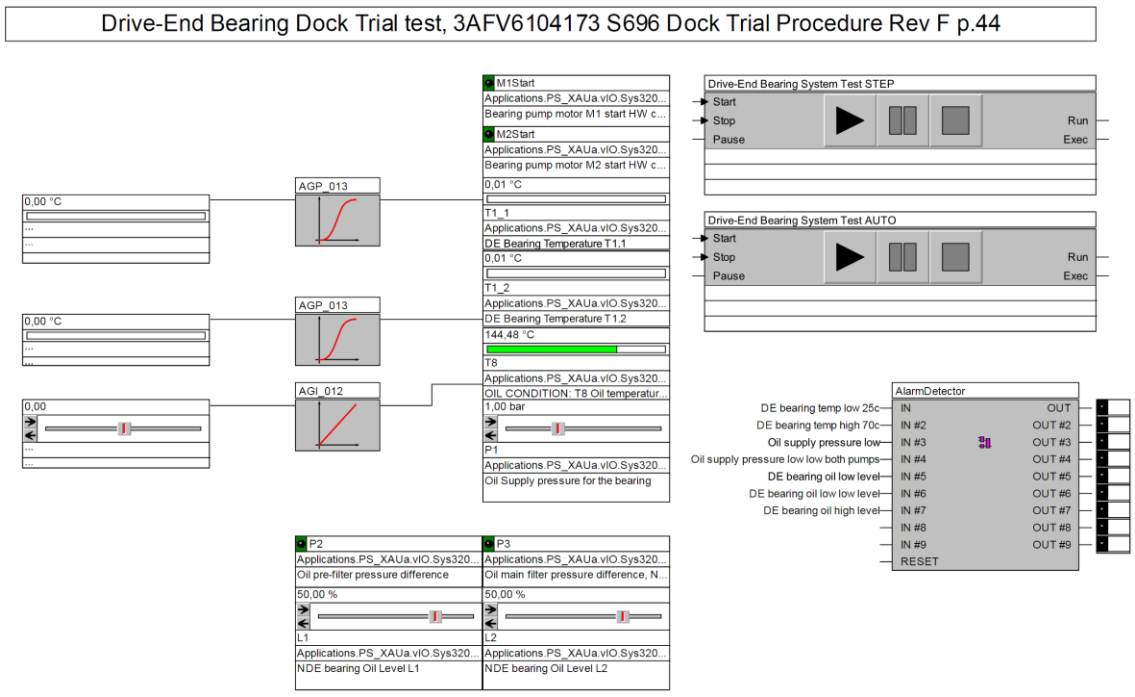


Figure 11. Drive-End Bearing test from Dock Trial test procedure.

Layout for the Temperature based protections test (Figure 12) for propulsion system is same as the previous test page with an exception of reduced functionalities and simpler simulation for analog signals. Test procedure verifies the system protections for

temperature-based events such as temperature rising to the critical level or alarm level. These system protections include alarms, power limitations, override functions and system interlocks which the AlarmDetector macro monitors during the testing.

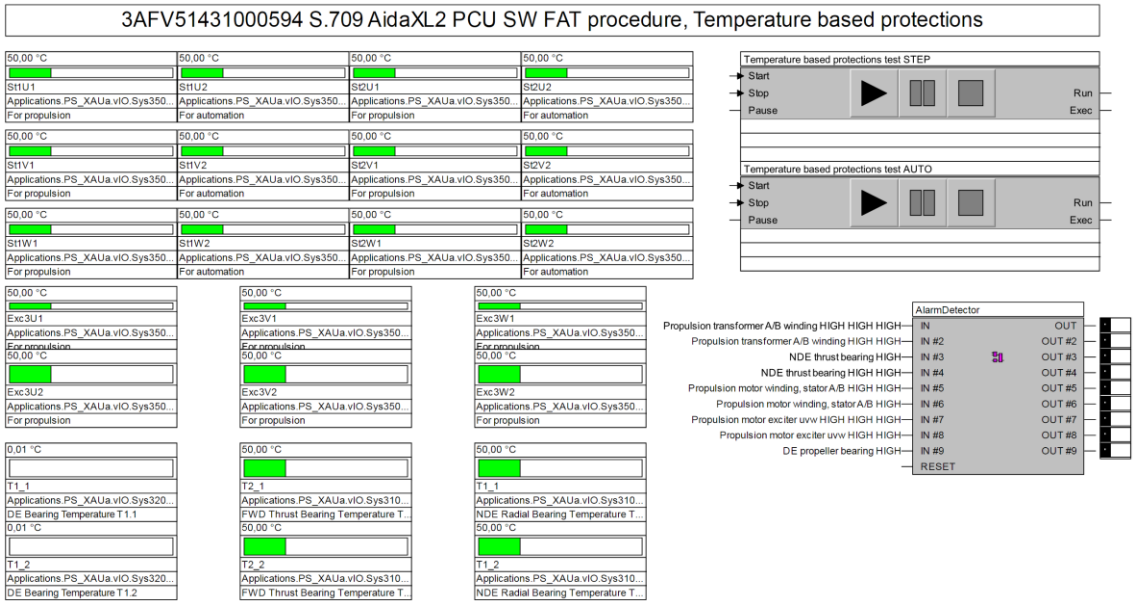


Figure 12. Temperature based protections test from PCU SW FAT procedure.

Auxiliary Devices duty/standby test (Figure 13) verifies the software for duty/standby control of the fans in the propulsion module. Objective is to test fan control for redundancy and changeover on device failure or process condition which in this case are automatic fan selection based on system parameters and user input (HMI). AIU is responsible for alarms, fan motor controls and master changeover in a fault situation (fan trips or is stopped). Purpose of this test was to study if WinMOD could replicate human inputs through HMI. In this case these HMI inputs were routed via OPC server which enabled simulating those using WinMOD.

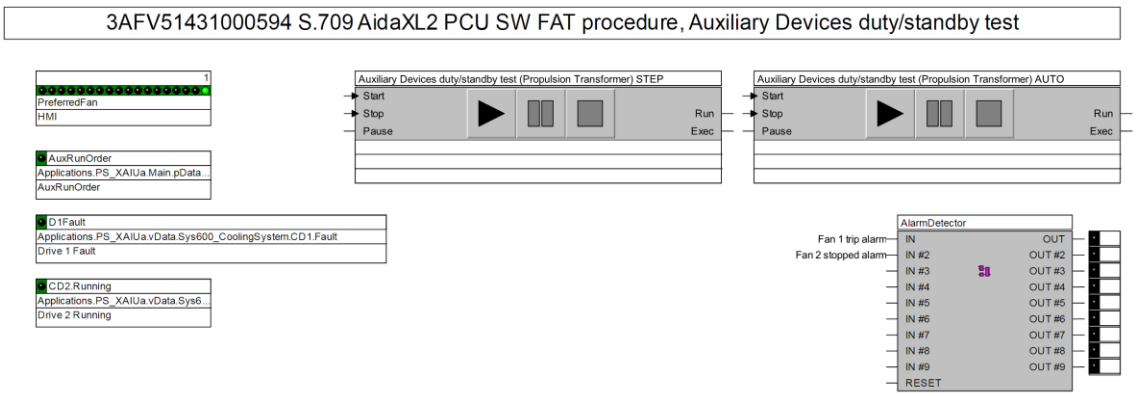


Figure 13. Auxiliary Devices duty/standby from test PCU SW FAT procedure.

6.2 Observing Test Results

AlarmDetector macro is used to monitor triggered AIU alarms. Indicator of the alarm tags changes color according to the status of alarm, where grey indicates untriggered alarm and blue indicates triggered alarm. Example of untriggered and triggered alarms can be seen in Figure 14.

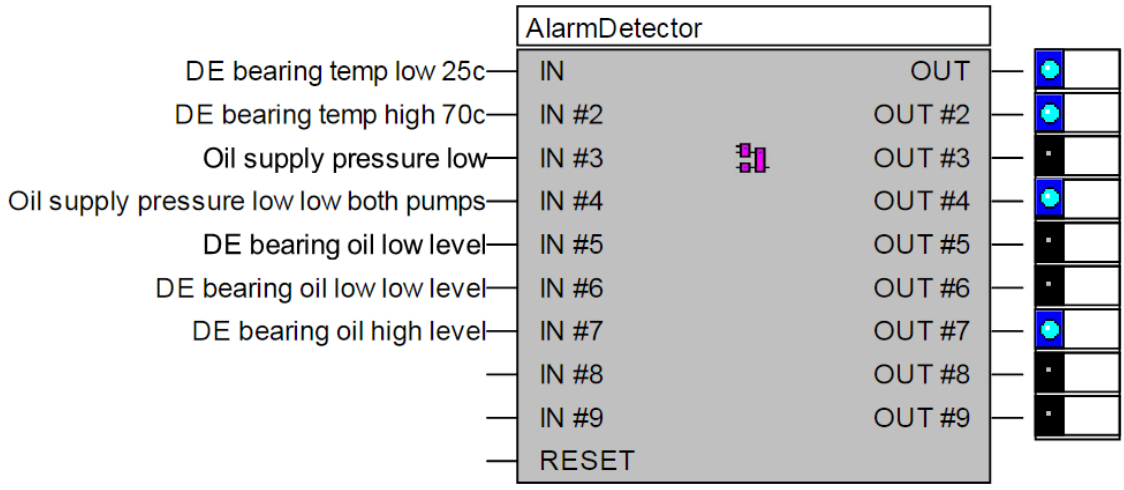


Figure 14. Untriggered and triggered alarms in AlarmDetector macro.

6.3 Examples of Individual System Pages

Total of ten individual system simulation pages were created for the simulator. Some of the systems were more complex than others, as an example the Sys330 Shaft Water

Seal OAW module (Figure 15). Some systems had less functionalities resulting a simpler simulation as can been seen in Figure 16 of Sys310 Non-Drive-End Bearing system. Some of the system functionalities required some specific signal processing such as impulse, ramp or step functions which were implemented by using corresponding simulation elements from WinMOD library. For functionalities which required even more complex behavior Analog Signal Simulation -macro was used as were discussed in Chapter 5.

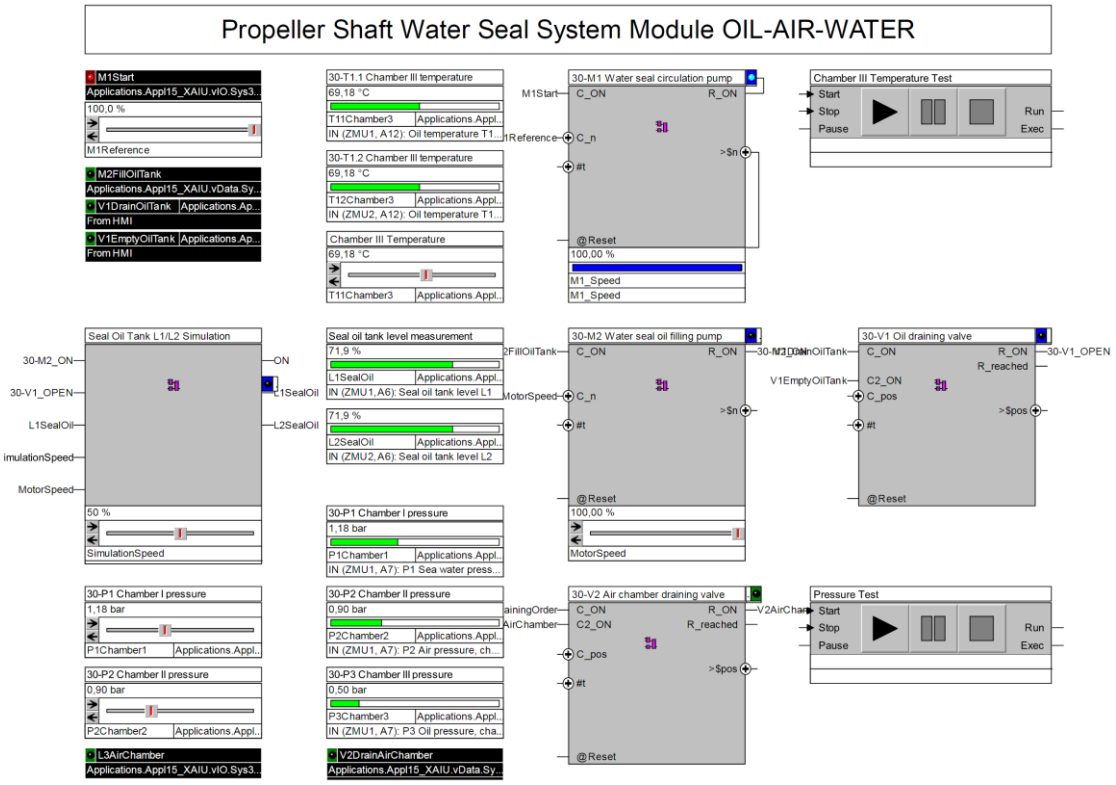


Figure 15. Analog Signal Simulation -macro is utilized as oil tank simulation in Propeller Shaft Seal page.

For simpler simulations basic elements were competent enough as they enabled signal forcing which was the only requirement for most of the functions. All the necessary signal types were easily forcible using the signal element itself. In Figure 16 three different types of signal elements (analog, binary and digital) were able to simulate the Non-Drive-End) Bearing System module as the module consisted only of these signals and required no other functionalities. Signal element was forcible through the user interface and WinMOD script.

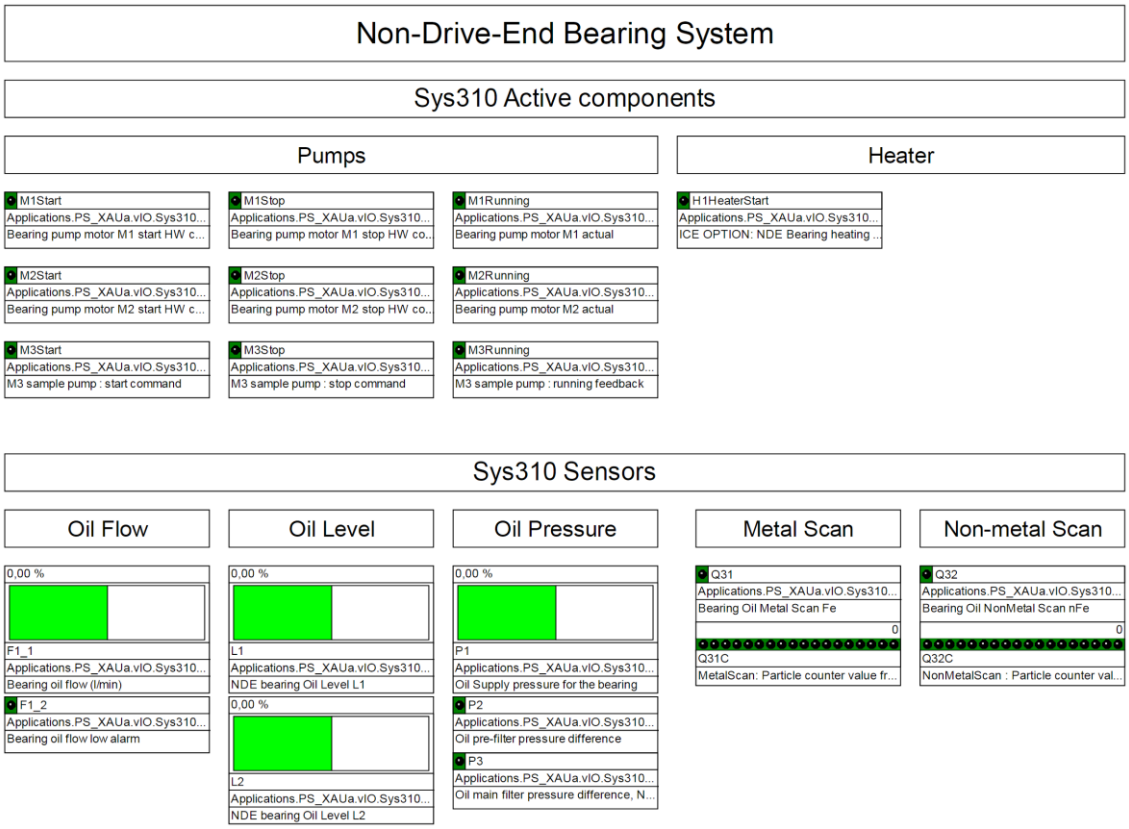


Figure 16. NDE (Non-Drive-End) Bearing System page.

6.4 WinMOD Scripts as a Test Procedure

WinMOD Simulator runs the testing script for AIU and returns it to a normal state after the test is done. Testing program consists of control commands and scripts which are delivered to the AIU in predefined order. Scripts consist of forcing commands, value changes and step comments. Total of three test scripts were written for this thesis work, Drive-End Bearing Dock Trial test procedure are presented in Listing 3. In this test the Azipod Drive-End Bearing system is tested by simulating changes in oil temperatures, oil pressures and oil levels in the bearing module. Script uses basic WinMOD script commands to force values for the signals and to print current step as text in the Force Machine. Scripts for the Temperature Based Protections test and Auxiliary Devices Duty/Standby test can be found in the Appendixes 1-2.

```
#WinMOD-Script V1.0
text "Starting DE Bearing oil lubrication"
cmd   "M1Start"  F=1
cmd   "M2Start"  F=1
cmd   "M1Start"  V=1
cmd   "M2Start"  V=1
```

```
wait 2000
text "DE bearing temperature low (T1 < 25 °C)"
cmd  "T1_1F"    F=1
cmd  "T1_2F"    F=1
cmd  "T1_1F"    V=25
cmd  "T1_2F"    V=25
wait 20000
text "DE bearing temperature high (T1 > 70 °C)"
cmd  "T1_1F"    V=70
cmd  "T1_2F"    V=70
wait 20000
text "Oil supply pressure low (< 0.7 Bar)"
cmd  "P1" V=0.70
wait 2000
text "Oil supply pressure low low (< 0.25 Bar)"
cmd  "P1" V=0.25
wait 2000
text "Oil supply pressure normal (1 Bar)"
cmd  "P1" V=1.0
wait 2000
text "DE bearing oil low level (< 29%)"
cmd  "L1" V=29
cmd  "L2" V=29
wait 2000
text "DE bearing oil low low level (< 26%)"
cmd  "L1" V=26
wait 2000
text "DE bearing oil high level (> 75%)"
cmd  "L1" V=75
wait 2000
text "DE bearing normal level (50%)"
cmd  "L1" V=50
cmd  "L2" V=50
wait 2000
text "Normal mode"
cmd  "M1Start" V=0
cmd  "M2Start" V=0
cmd  "M2Start" F=0
cmd  "M1Start" F=0
```

Listing 3. Drive-End Bearing Dock Trial Test Procedure script in WinMOD.

6.5 Macros

In Figure 17 operation of Analog Signal Simulation macro is shown. Macro element consists of few essential mathematical functions and three boolean operators for added functionality of the macro element. State of the boolean operators Pump and Valve in Figure 17 activates and deactivates the circuit while the output S_ON indicates the state of the simulation. Simulation speed and scale can be adjusted using operators SimulationSpeed and DeltaA which both affects to different factors in the AFF_015 function block which houses the mathematical function responsible for signal simulation.

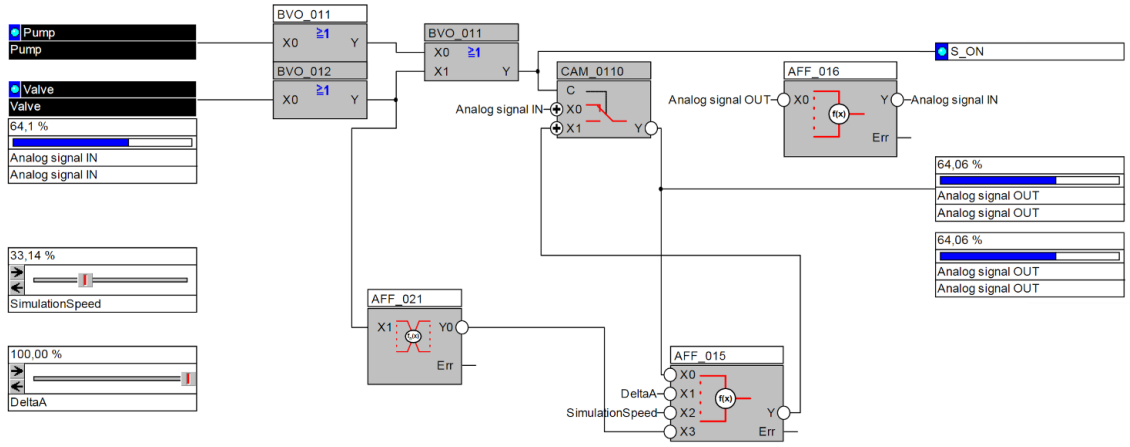


Figure 17. Analog Signal Simulation as Oil Tank simulation macro element.

Alarm Detector Macro element in Figure 18 is responsible of monitoring the AIU signals. In Figure 18 the principle of the macro element is shown. Monitoring is based on SET/RESET functions which are triggered instantly if the monitored signal activates. Single macro element is able to monitor nine different signals. In cases where more signals need to be monitored, multiple macro elements can be used.

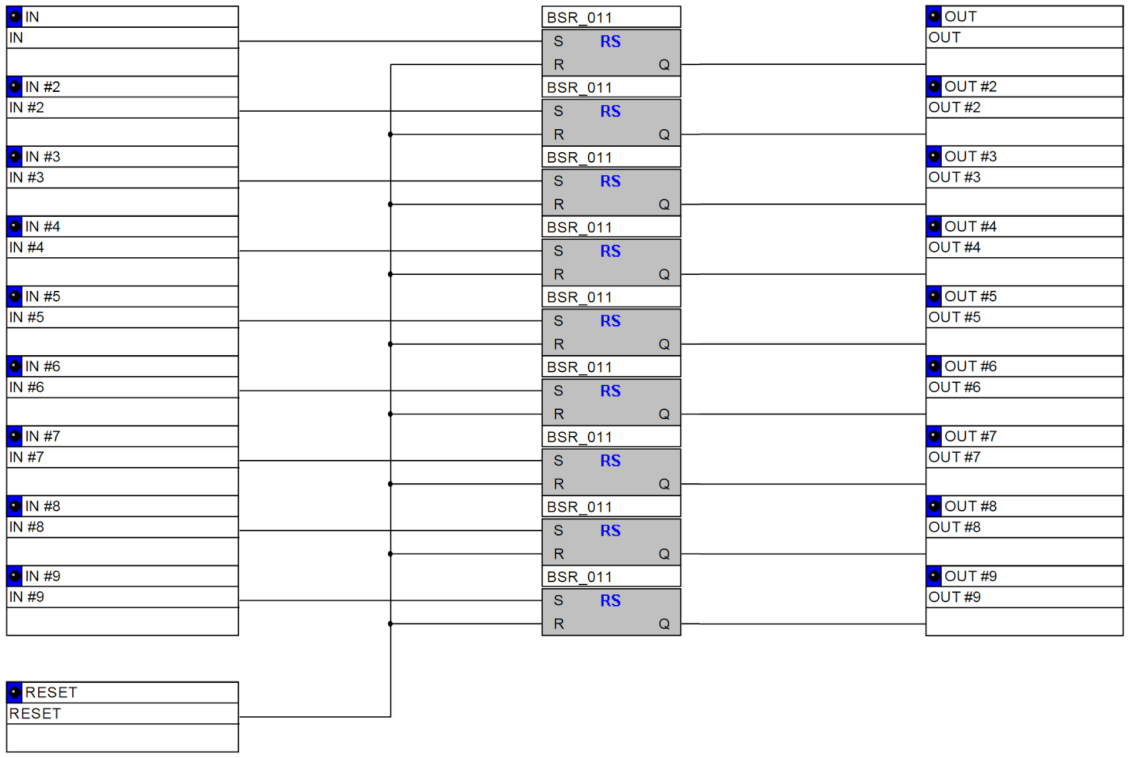


Figure 18. Alarm Detector macro.

7 Conclusion

Objective of this thesis work was to build simulation platform for ABB AIU X product and to study suitability of WinMOD software as an internal testing tool for ABB Marine & Ports. As a result of the thesis work, ABB has now the option to use WinMOD simulation testing environment to perform a configurable IAT for Azipod X AIU at any location and at any instant of time. Test and simulator can be configured without deeper knowledge of the programs code and there is possibility to extend the IAT to include other Azipod products. The constraints imposed by the software and hardware have been recognized and simulation platform has been built within these limits. The suitability of the WinMOD for ABB's testing purposes has been evaluated and the concept of AIU virtual testing has been studied.

8 Summary

In this thesis AIU X hardware-in-the-loop simulator were built to streamline the testing procedure of AIU X product. As a result, it is now possible to use this simulator as a testing tool for the ABB AIU X product. Thesis work successfully completed the proof of concept, which included the study of WinMODs suitability as a testing tool and mapping of the simulator capabilities. Focus of the simulator was to verify AIU auxiliary control software by executing key points from FAT and SAT test procedures which was major part of the proof of concept. Simulator scalability and adaptability were studied and the objectives for the thesis work were achieved.

Total of three different test procedures were created for the simulator to demonstrate the capabilities of the simulator. These test procedures can be easily broadened to include more test items from any test procedures. It is also possible to develop the WinMOD simulator concept further to include other Azipod products. Simulator at its current state can be used as preliminary testing tool but using it as standardized and reliable testing tool it needs further development due to challenges in OPC communication.

References

1 Norwegian University Of Science And Technology. Marine HIL simulation laboratory (HIL-lab) [online]. URL: https://www.ntnu.edu/imt/lab/hil. Accessed 17 February 2020.

2 MathWorks. Hardware-in-the-Loop (HIL) Simulation [online]. URL: https://www.mathworks.com/discovery/hardware-in-the-loop-hil.html. Accessed 17 February 2020.

3 MathWorks. What Is Hardware-In-The-Loop Simulation? [online]. URL: https://www.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html. Accessed 17 February 2020.

4 MathWorks. Perform Hardware-in-the-Loop Simulation with MATLAB and Simulink to Test and Validate Control Algorithms [online]. 2016. URL: https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/s/89952_93049v00_Simulink_Real-Time_Whitepaper.pdf. Accessed 17 February 2020.

5 Fakir; Brunini; Godoy. Hardware In The Loop Simulation For Industrial Process Control [online]. Group of Automation And Integration Systems (GASI), São Paulo State University, 2014. URL: https://2ae7us20z1th3cu2bkjoza11-wpengine.netdna-ssl.com/wp-content/uploads/2014/11/Hardware-in-the-Loop-Simulation-for-Industrial-Process-Control.pdf. Accessed 17 February 2020.

6 Hendrickson, Chris. The Design and Construction Process [online]. Carnegie Mellon University, 2008. URL: https://www.cmu.edu/cee/projects/PMbook/03_The_Design_And_Construction_Process.html#3.12 Computer-Aided Engineering. Accessed 19 February 2020.

7 DNV GL. HIL Testing Concept Explanation [online]. URL: https://www.dnvgl.com/services/hil-testing-concept-explanation--83385. Accessed 18 February 2020.

8 Maloney, Peter. Using Hardware in the Loop to Create a Microgrid Testbed [online]. 2019. URL: https://microgridknowledge.com/microgrid-testbeds-hil-simulation/. Accessed 19 February 2020.

9 Bivens, H.P.; Williamson, G.A. Jr.; Luger, G.F.; Erdmann, R.G.; Maguire, M.C.; Baldwin, M.D. et al. WinMod: An expert advisor for investment casting [online]. 1998 URL: https://www.aaai.org/Papers/SIGMAN/1998/SIGMAN98-003.pdf. Accessed 18 February 2020.

10 ABB. Integrated Marine Systems Lab – electrical and automation systems testbed. [online]. URL: https://search.abb.com/library/Download.aspx?DocumentID=9AKK107045A7593&LanguageCode=en&DocumentPartId=&Action=Launch. Accessed 20 February 2020.

11 National Instruments. Hardware-in-the-Loop (HIL) Test [online]. URL: https://www.ni.com/fi-fi/innovations/automotive/hardware-in-the-loop.html. Accessed 20 February 2020.

12 Mewes & Partner GmbH. What is WinMOD? [online]. URL: https://www.winmod.de/english/. Accessed 20 February 2020.

13 Controllab Products. WinMOD [online]. URL: https://www.hil-simulation.com/technology/hil-simulator/winmod.html. Accessed 20 February 2020.

14 Mewes & Partner GmbH. WinMOD System Software [online]. URL: https://www.winmod.de/english/products/winmod-systemsoftware/. Accessed 20 February 2020.

15 Mewes & Partner GmbH. Virtual Commissioning [online]. URL: https://www.winmod.de/english/utilizations/virtual-commisioning. Accessed 17 February 2020.

16 PassMark Software. CPU Benchmarks [online]. URL: https://www.cpubenchmark.net/compare/Intel-i7-8665U-vs-Intel-i5-4300U/3434vs2054. Accessed 21 February 2020.

17 Mewes & Partner GmbH. WinMOD System Requirements [online]. URL: https://www.winmod.de/english/products/winmod-systemsoftware/systemvoraussetzungen/. Accessed 21 February 2020.

18 Mewes & Partner GmbH. WinMOD System Software 7.2 Manual. 2018. Mewes & Partner GmbH.

19 ABB. Azipod® Product Platform Selection Guide [online]. 2010. URL: https://search-ext.abb.com/library/Download.aspx?DocumentID=9AKK105152A5238&LanguageCode=en&DocumentPartId=&Action=Launch. Accessed 21 February 2020.

20 ABB Marine and Ports. Azipod XO2300, 3AFV6027876 − en, rev. M. 2017. ABB.

21 ABB Marine and Ports. Azipod Interface Unit, Function Description, 3AFV6021446, rev. D. 2016. ABB.

22 ABB. Azipod® XO2100 and XO2300 Product Introduction [online]. 2012. URL: https://search-ext.abb.com/library/Download.aspx?DocumentID=3AFV6016618&LanguageCode=en&DocumentPartId=&Action=Launch. Accessed 17 February 2020.

23 Mewes & Partner GmbH. AC 800 Specs, WinMOD Configuration Y500 V7.2 E. 2018. Mewes & Partner GmbH.

AIUX_PCU_SW_FAT_TemperatureBasedProtections Test Script

```
#WinMOD-Script V1.0

Text "Propulsion transformer A/B, Winding temperature"
cmd  "St1U1"    F=1
cmd  "St1U2"    F=1
cmd  "St2U1"    F=1
cmd  "St2U2"    F=1
wait 200
Text "Temperature High 130C"
cmd  "St1U1"    V=130
cmd  "St1U2"    V=130
cmd  "St2U1"    V=130
cmd  "St2U2"    V=130
wait 10000
Text "Temperature Power limit 135C"
cmd  "St1U1"    V=135
cmd  "St1U2"    V=135
cmd  "St2U1"    V=135
cmd  "St2U2"    V=135
wait 10000
Text "Temperature High High 130C"
cmd  "St1U1"    V=140
cmd  "St1U2"    V=140
cmd  "St2U1"    V=140
cmd  "St2U2"    V=140
wait 10000
Text "Temperature trip 145C"
cmd  "St1U1"    V=145
cmd  "St1U2"    V=145
cmd  "St2U1"    V=145
cmd  "St2U2"    V=145
wait 10000
Text "Temperature reset"
cmd  "St1U1"    V=50
cmd  "St1U2"    V=50
cmd  "St2U1"    V=50
cmd  "St2U2"    V=50
wait 10000
Text "Force reset"
cmd  "St1U1"    F=0
cmd  "St1U2"    F=0
cmd  "St2U1"    F=0
cmd  "St2U2"    F=0
wait 200
Text "Propulsion motor winding, stator A/B"
cmd  "St1V1"    F=1
cmd  "St1V2"    F=1
cmd  "St2V1"    F=1
cmd  "St2V2"    F=1
wait 200
Text "Temperature High 140C"
cmd  "St1V1"    V=140
cmd  "St1V2"    V=140
cmd  "St2V1"    V=140
cmd  "St2V2"    V=140
wait 10000
Text "Temperature Power limit 145C"
cmd  "St1V1"    V=145
cmd  "St1V2"    V=145
cmd  "St2V1"    V=145
cmd  "St2V2"    V=145
```

```
wait 10000
Text "Temperature High High 150C"
cmd  "St1V1"    V=150
cmd  "St1V2"    V=150
cmd  "St2V1"    V=150
cmd  "St2V2"    V=150
wait 10000
Text "Temperature trip 155C"
cmd  "St1V1"    V=155
cmd  "St1V2"    V=155
cmd  "St2V1"    V=155
cmd  "St2V2"    V=155
wait 10000
Text "Temperature Reset"
cmd  "St1V1"    V=50
cmd  "St1V2"    V=50
cmd  "St2V1"    V=50
cmd  "St2V2"    V=50
wait 10000
Text "Force Reset"
cmd  "St1V1"    F=0
cmd  "St1V2"    F=0
cmd  "St2V1"    F=0
cmd  "St2V2"    F=0
wait 200
Text "Propulsion motor winding A/B"
cmd  "St1W1"    F=1
cmd  "St1W2"    F=1
cmd  "St2W1"    F=1
cmd  "St2W2"    F=1
wait 200
Text "Temperature High 140C"
cmd  "St1W1"    V=140
cmd  "St1W2"    V=140
cmd  "St2W1"    V=140
cmd  "St2W2"    V=140
wait 10000
Text "Temperature Power limit 145C"
cmd  "St1W1"    V=145
cmd  "St1W2"    V=145
cmd  "St2W1"    V=145
cmd  "St2W2"    V=145
wait 10000
Text "Temperature High High 150C"
cmd  "St1W1"    V=150
cmd  "St1W2"    V=150
cmd  "St2W1"    V=150
cmd  "St2W2"    V=150
wait 10000
Text "Temperature trip 155C"
cmd  "St1W1"    V=155
cmd  "St1W2"    V=155
cmd  "St2W1"    V=155
cmd  "St2W2"    V=155
wait 10000
Text "Temperature Reset"
cmd  "St1W1"    V=50
cmd  "St1W2"    V=50
cmd  "St2W1"    V=50
cmd  "St2W2"    V=50
wait 10000
Text "Force Reset"
cmd  "St1W1"    F=0
cmd  "St1W2"    F=0
cmd  "St2W1"    F=0
```

```
cmd  "St2W2"    F=0
wait 200
Text "Propulsion motor exciter"
cmd  "Exc3U1"   F=1
cmd  "Exc3V1"   F=1
cmd  "Exc3W1"   F=1
wait 200
Text "Temperature High 140C"
cmd  "Exc3U1"   V=140
cmd  "Exc3V1"   V=140
cmd  "Exc3W1"   V=140
wait 10000
Text "Temperature Power limit 145C"
cmd  "Exc3U1"   V=145
cmd  "Exc3V1"   V=145
cmd  "Exc3W1"   V=145
wait 10000
Text "Temperature High High 150C"
cmd  "Exc3U1"   V=150
cmd  "Exc3V1"   V=150
cmd  "Exc3W1"   V=150
wait 10000
Text "Temperature trip 155C"
cmd  "Exc3U1"   V=155
cmd  "Exc3V1"   V=155
cmd  "Exc3W1"   V=155
wait 10000
Text "Temperature Reset"
cmd  "Exc3U1"   V=50
cmd  "Exc3V1"   V=50
cmd  "Exc3W1"   V=50
wait 10000
Text "Force Reset"
cmd  "Exc3U1"   F=0
cmd  "Exc3V1"   F=0
cmd  "Exc3W1"   F=0
wait 200
Text "DE (Propeller) Bearing"
cmd  "T1_1F"    F=1
cmd  "T1_2F"    F=1
wait 200
Text "Temperature High 70C"
cmd  "T1_1F"    V=70
cmd  "T1_2F"    V=70
wait 10000
Text "Temperature Power limit 80C"
cmd  "T1_1F"    V=80
cmd  "T1_2F"    V=80
wait 10000
Text "Temperature High High 90C"
cmd  "T1_1F"    V=90
cmd  "T1_2F"    V=90
wait 10000
Text "Temperature trip 100C"
cmd  "T1_1F"    V=100
cmd  "T1_2F"    V=100
wait 10000
Text "Temperature Reset"
cmd  "T1_1F"    V=50
cmd  "T1_2F"    V=50
wait 10000
Text "Force Reset"
cmd  "T1_1F"    F=0
cmd  "T1_2F"    F=0
wait 200
```

```
Text "NDE (Thrust) Bearing"
cmd  "T2_1"     F=1
cmd  "T2_2"     F=1
wait 200
Text "Temperature High 80C"
cmd  "T2_1"     V=80
cmd  "T2_2"     V=80
wait 10000
Text "Temperature Power limit 90C"
cmd  "T2_1"     V=90
cmd  "T2_2"     V=90
wait 10000
Text "Temperature High High 95C"
cmd  "T2_1"     V=95
cmd  "T2_2"     V=95
wait 10000
Text "Temperature trip 100C"
cmd  "T2_1"     V=100
cmd  "T2_2"     V=100
wait 10000
Text "Temperature Reset"
cmd  "T2_1"     V=50
cmd  "T2_2"     V=50
wait 10000
Text "Force Reset"
cmd  "T2_1"     F=0
cmd  "T2_2"     F=0
wait 200
Text "NDE (Radial) Bearing"
cmd  "T1_1R"    F=1
cmd  "T1_2R"    F=1
wait 200
Text "Temperature High 70C"
cmd  "T1_1R"    V=70
cmd  "T1_2R"    V=70
wait 10000
Text "Temperature Power limit 80C"
cmd  "T1_1R"    V=80
cmd  "T1_2R"    V=80
wait 10000
Text "Temperature High High 90C"
cmd  "T1_1R"    V=90
cmd  "T1_2R"    V=90
wait 10000
Text "Temperature trip 100C"
cmd  "T1_1R"    V=100
cmd  "T1_2R"    V=100
wait 10000
Text "Temperature Reset"
cmd  "T1_1R"    V=50
cmd  "T1_2R"    V=50
wait 10000
Text "Force Reset"
cmd  "T1_1R"    F=0
cmd  "T1_2R"    F=0
```

Auxiliary Devices Duty/Standby Test Script

```
#WinMOD-Script V1.0

text "Selecting Fan 1 as a master"
cmd  "PreferredFan"  F=1
cmd  "PreferredFan"  V=1
wait 5000
text "Starting Aux"
cmd  "AuxRunOrder"   F=1
cmd  "AuxRunOrder"   V=1
wait 5000
text "Stopping Aux"
cmd  "PreferredFan"  F=0
cmd  "AuxRunOrder"   V=0
wait 5000
text "Starting Aux, Fan 2 should start."
cmd  "AuxRunOrder"   V=1
wait 5000
text "Selecting Fan 1 as a master"
cmd  "PreferredFan"  F=1
cmd  "PreferredFan"  V=1
wait 5000
text "Tripping of Fan 1"
cmd  "D1Fault"  F=1
cmd  "D1Fault"  V=1
wait 5000
text "Recovering Fan 1"
cmd  "D1Fault"  V=0
cmd  "D1Fault"  F=0
wait 5000
text "Selecting Fan 2 as a master"
cmd  "PreferredFan"  V=2
wait 5000
text "Simulating stopping Fan 2"
cmd  "CD2.Running"   F=1
cmd  "CD2.Running"   V=0
wait 5000
text "Reset Fan 2 state"
cmd  "CD2.Running"   F=0
wait 5000
text "Reset test"
cmd  "PreferredFan"  F=0
cmd  "AuxRunOrder"   F=0
```