WIN GD
A smarter perspective

CIMAC 19 CONGRESS
VANCOUVER, JUNE 10-14, 2019

WiCE - New Engine Control System for Two-Stroke Engines

3 - Electronic Support - Controls, Automation, Measurement & Monitoring

Paper 165

Wolfgang Östreicher, WinGD
General Manager
Automation & Control

Alexander Doncic, WinGD
Team Leader, Requirements & SW Deployment

Stefan Goranov, WinGD
Program Manager, Hybridisation

Andrew Pennycott, WinGD
Project Manager, Automation & Control

Maria Signorile, WinGD
Team Leader, Software Development

Special thanks to contributors:
Robert Hunger, SCS Super Computing Systems
Christof Sidler, SCS Super Computing Systems

ABSTRACT

A new control system is under development at Winterthur Gas & Diesel to address the rising demands on the control aspects and diagnostic capabilities of digital engine control systems. Key features include a modular design of the hardware components that, for the 2-stroke engine, include the MCU (main control unit), CCU (cylinder control module) and GTU (gateway unit). In addition to well-known connections such as SSI and bus systems (CAN), the much more powerful Ethernet bus structure will be applied. With the installation of redundant Ethernet rings, a data collection platform suitable for far more powerful monitoring and diagnostic systems has been realised.

The software architecture is divided into layers according to state-of-the-art concepts. The system software, including all elements of the software foundation, is newly developed and runs on a Real Time Operating System (RTOS with Linux kernel).

Using graphical development tools (MATLAB/Simulink®), the transfer of modularized application software from existing control applications to the new control hardware reuses existing experience, including user and customer feedback, to improve engine control and to facilitate commissioning and operation.

Also, tools for commissioning are selected with best user and customer feedback and are adapted to the new hardware and system software by taking modern GUI features.

Furthermore, as this development is part of an CSSC innovation program, the architecture is fully modular which makes the technical solutions easily adaptable for future needs in the marine environment.

1 INTRODUCTION

Electronic engine control systems on combustion engines were developed and introduced in the last century. Meanwhile, this technology has become well accepted, and indeed, is now state-of-the-art. On large two-stroke engines that mainly drive commercial deep-sea ships, a different approach is inconceivable. The electronic approach has enabled optimization towards cleaner engines, with improvements in efficiency being delivered.

However, demands are rising and complexity is progressively increasing. These include the environmental need to continuously cut emissions with the long-term target of zero emissions [1,4] as well as the commercial need to further reduce the cost of transportation by reducing fuel consumption. These are in addition to requests of rising availability, predictability and serviceability of the machinery. New concepts, with new business models and scenarios especially based on the use of digital technology, are generating a multitude of interconnections and interactions that must be supported and secured, leading to requests for ever higher performance concerning the computation and communication capabilities of electronic control systems [2,3].

Given that the current engine control systems of WinGD (Winterthur Gas & Diesel) engines had been developed and launched more than 20 years ago, a new engine control platform to meet all the coming demands of the next decades has been developed. This new control platform is presented in this paper. An outline of the requirements of the new system is firstly provided, followed by details of the actual implementation and realization of the platform. Subsequently, the testing and deployment of the system are described.

2 REQUIREMENTS

The project for the development of a new engine control system (ECS) began in early 2016 with a short time-to-market. Development, especially concerning software, is subject to constraints such as standards, regulations, limited resources, the need for high autonomy, market competition etc. Therefore, the project has been executed according the V-Model as a customizable, state-of-the-art development process, enabling an efficient response to all the inherent challenges. The first steps comprised the elicitation and development of all the requirements for such a control system.

On the one hand, the system shall have the ability to meet future requirements such as sufficient computing power combined with modern hardware interfaces for the intended product life of 15 to 20 years. On the other hand, an expensive system and excessive required computing power would limit market impact.

The requirements for simple implementation, easy and incremental expansion and highest quality were considered. Therefore, a modular architecture is the first choice for the whole system: for the hardware (HW) as well as for the software (SW).

The engine control system must increasingly be able to connect to other electronic systems on board. Today, it is expected that a remote access to and from land be realized. This adds to the already high security requirements. With a modular approach, further requirements can be straightforwardly implemented by applying a plug-in strategy. Nevertheless, maintainability and operability remain in strong focus.

The users shall receive an easily understandable system such that the operators and crew are able to maintain the system. They also need to understand upcoming and existing problems. Moreover, they need a straightforward means of analyzing and diagnosing system failures. A remote access to the system for monitoring and troubleshooting supports these needs. Data collected by the control system can be remotely analyzed by experts, thus supporting the safety, reliability and availability of the system. These options have a positive impact on maintenance costs and safeguard investment.

The total cost of ownership is another important target. To enable a low-cost system, easy commissioning and efficient software configuration for all applications should be built-in. In future, remote software updates will also be possible. Modern control systems rely more and more on software and its continuously expanding data. Consequently, automated configuration and tuning must feature as core features of these applications.

Finally, the increasing number of functions and capabilities of the engine control system must be supported by efficient software development, maintenance and automated test functions.

3 THE NEW CONTROL SYSTEM

The new control system generation developed – WinGD integrated Control Electronics (“WiCE”) – is integrated into the propulsion system as

displayed in Figure 1. Firstly, the “firewall”, an integral part of WiCE, is visible. This new unit provides universally-secured connectivity. Moreover, it has sufficient computing power and resources to acquire, buffer and pre-process fast-sampled signals from the control system.

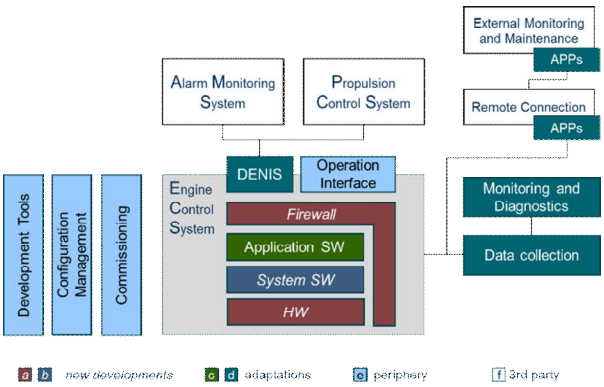

Figure 1. Engine control system logically integrated in propulsion system.

Secondly, behind the firewall, there are connections for external data collection and diagnostic systems such as WinGD’s WiDE [6] and further systems which can be integrated. The well-known connections to a vessel’s propulsion control system (PCS) as well as the alarm and monitoring system (ALM) are kept and organized by the DENIS interface as specified in the past or the WECS and UNIC control systems.

3.1 System

Figure 2 visualizes the WiCE system topology. In contrast to previous control system architectures of WinGD, all the modules are connected via two physically separate Ethernet buses in a ring topology. One physical ring is exclusively used for transmission of all essential control data that are to be sent redundantly between all modules, including the absolute crank angle signal. The other ring transmits non-essential data that have no redundancy requirements. Examples are diagnostic data from the control modules to the monitoring systems and system software services like downloads or parameter synchronization.

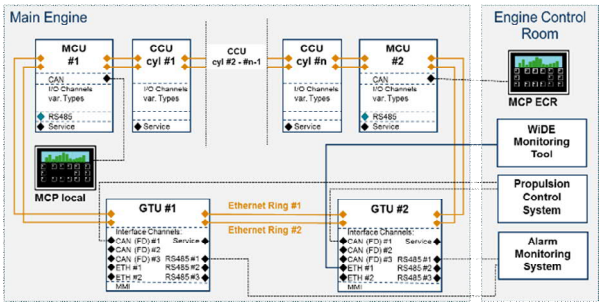

Figure 2. System topology for 2-stroke engines

The system bus is not necessarily a bus in the strict sense, but rather is a more general communication network. The system bus provides sufficient bandwidth to transport the arising traffic. This corresponds to periodic traffic used for the exchange of control data, occasional traffic for tasks like software updates, and monitoring of engine states and testing. It also ensures a reliable low-jitter and low-latency distribution of the engine’s crank-angle signal.

The system bus is intended to interconnect the electronic hardware units only. Any peripheral system must not be connected to the ECS internal system bus, but instead to the firewall.

The network connects several control modules of the ECS; more details are provided later in this section. Common to these modules in the system is their architecture, which is organized in a modular structure, Figure 3, with the usual three main layers for hardware, basic software or foundation and application software. The hardware and the application software are specifically arranged so that each control module type complies best with the main tasks it must perform; however, the foundation or system software is common for all modules.

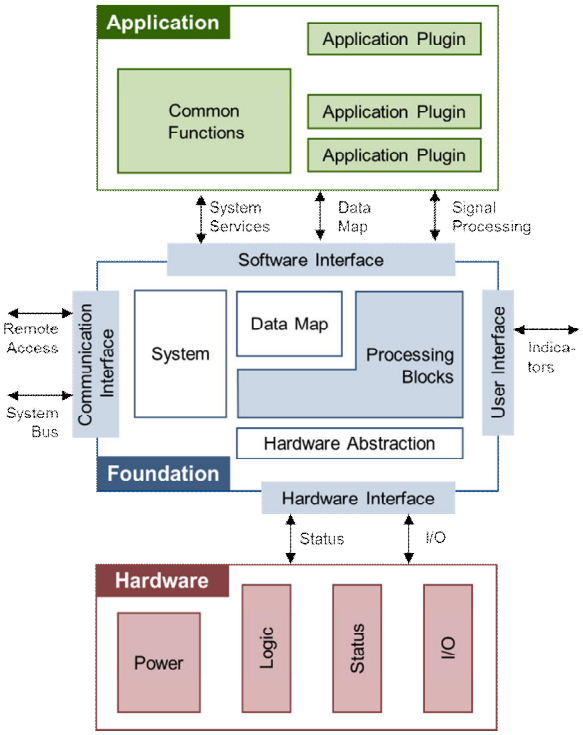

Figure 3. Modular control structure

3.2 Hardware

The hardware contains the logic for the control board, the inputs and outputs, the necessary power supply and status information. The architecture of the hardware again follows a generic and modular set-up so that a given control module can easily be configured - see Figure 4.

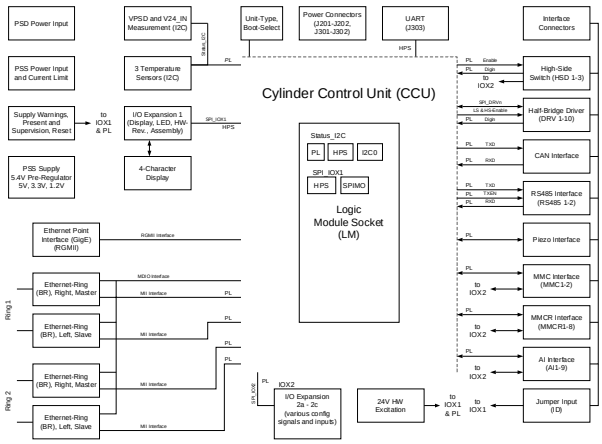

Figure 4: Structure of the cylinder control module with the modules made

3.2.1 Logic Subsystem

The logic components of the hardware are combined on a logic module (LM), which is common for all module types. As the core of this module a SoC (system on chip) has been selected that includes a dual core CPU (Central Processing Unit) and the FPGA (Field Programmable Gate Array). Moreover, the volatile memory, as RAM (Random-Access Memory), and the non-volatile memory, as flash memory, are located on the LM.

3.2.2 Modules and Interfaces

Increased diversification is needed for the more specific control modules. However, the modules all are designed using off-the-shelve solutions as submodules that are designed for specific tasks:

- The first part of the power subsystem is the **Power Supply Driver (PSD)**, which provides the half bridge drives with the necessary energy. In addition to the well dimensioned capacitors that ensure sufficient energy at all stages, this module includes overvoltage protection at the input and a means to suppress spikes on every in and output.
- The second part of the power subsystem is the **Power Supply System (PSS)** provides energy to the system logic all the other interfaces, except the halve bridge drivers.
- The **HW controlled excitation (12 V HW)** that simply delivers up to 500 mA from the 24 V internal supply rail to an external load.
- The aforementioned **Half Bridge Driver (DRV** capable of switching currents of a continuous 5 A level and peak currents of up to 25 A. Open-load detection is integrated as well as overcurrent limitation and a limitation of voltage spikes from inductive loads.
- **High Side Driver (HSD)**, capable of driving up to 5 A continuous power with overcurrent protection and status monitoring.
- **Analog Input (AI)** for 4-20 mA inputs with overload and over-current (30 mA) protection.
- **Multi-Mode Channel (MMC)**, configurable for both analog inputs and outputs, and digital inputs and outputs, and has a built-in PT1000 interface temperature sensor.
- **Multi-Mode Isolated (MMI)**, configurable for isolated digital inputs and outputs, as well as isolated analog inputs analog outputs.
- **Identity Input (ID)** for the module ID signal.
- **Piezo Sensor Interface** is a module for reading and preprocessing of the signals of a piezo sensor.
- **CAN** Interface
- **RS485** Interface
- **Ethernet (Point)** Interfaces, with automatic speed selection of 10/100/1000 Mbps.
- **Ethernet (Ring)** Interfaces, the module for the system bus and communication network
- **USB** Interface

This modular set-up makes it possible to combine and assemble all selected subsystems on an evaluation board, Figure 5, for performing basic tests in an early development stage.

Figure 5: Evaluation Board

This evaluation board has been created for early proofing design as well as constructing and testing different project-specific concepts.

3.2.3 Control Modules

The electronic hardware of two-stroke engines' ECS in detail is composed by the following three types of control modules physically installed on the engine.

The main control unit (MCU), Figure 6, is the central device in the ECS and is responsible for carrying out tasks for the engine as a whole and thus regularly provides, via the communication network, all the other modules with collected data on the state of the engine.

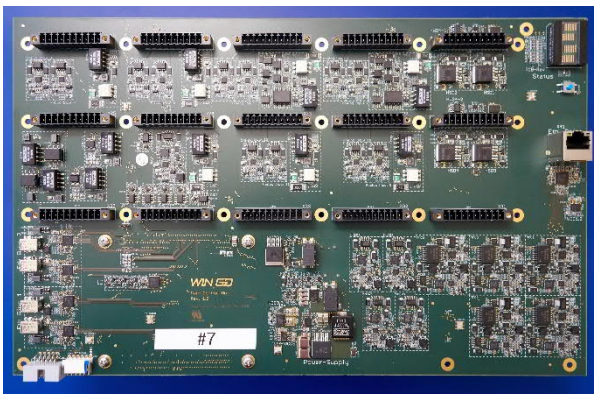

Figure 6: WiCE - Main Control Unit (MCU)

The cylinder control unit (CCU), Figure 7, drives the actuator installed on each cylinder of an engine (one is mounted per cylinder) and runs the fast local cylinder control loops, where it drives all the hardware outputs and processes all the cylinder individual sensor inputs. As with the MCU, the CCU is connected to the communication network to exchange data with the other modules.

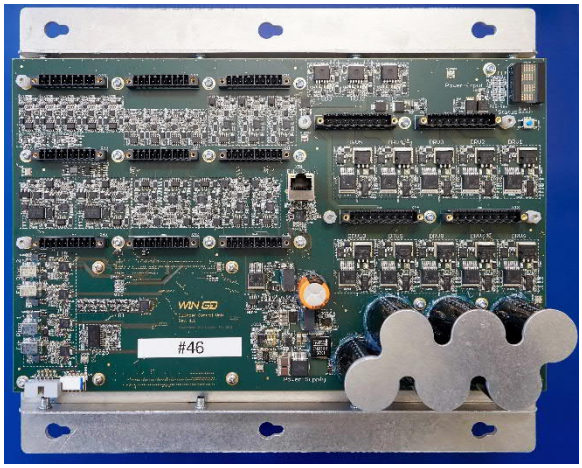

Figure 7: WiCE - Cylinder Control Unit (CCU)

The gateway unit (GTU), Figure 8, is the "firewall" that allows the ECS to communicate with the engine-external systems; indeed, it is the only way for these systems to interact. It fulfills security tasks and converts between different communication protocols that the ECS can process and also provides configurable input measurement, analogue current output, and digital input and output.

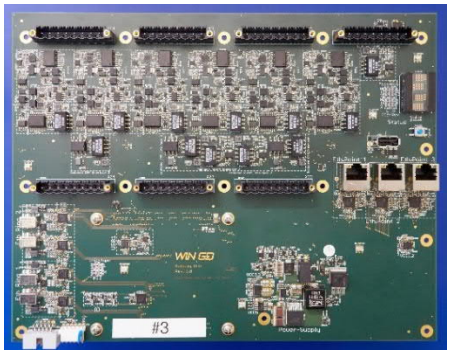

Figure 8: WiCE - Gateway Unit (GTU)

The interfaces between the user and the control system and the manual control panels (MCP) are mounted on the engine as a separate box alongside the assembled interface for the propulsion control system, and in the engine control room, Figure 9. Each is linked to one MCU.

Figure 9: WiCE – Manual Control Panel (MCP) with touch screen and fuel dial.

3.2.4 Hardware testing

A stable and robust hardware is the base for high availability. Requirements are elicited to ensure best quality. According these requirements all modules were tested.

The tests for the marine-type approvals are specified by the classification societies and standards for control systems. Typical testing performed includes evaluation of tolerance to heat and humidity, performance in cold conditions, vibration and shock tests, flammability tests, power supply failure and insulation resistance testing, high voltage evaluation, and electromagnetic compatibility (EMC) checks.

For ensuring quality of the products in the field, all individual boards are to be tested after production and before assembly to check that the modules are functioning correctly. In the same compartment, the boot loader, MAC address (Media Access Control), serial number etc. are written to the logic module.

3.3 Foundation

For development of the foundation, the following design concepts are applied:

- **Communication via the System Bus**: All components are connected to a single communication network that utilizes broadcast messages to exchange information.
- **Modular Design and Separation of Function Categories**: Modularization is used extensively to structure and break down the complex tasks at hand. This simplifies development, testing and maintenance of the individual components.
- **Design for Redundancy**: Implementing critical modules redundantly makes the system tolerant to single-module failures. When a module enters an erroneous state, the other modules can still provide the required functionality.
- **Layering Principle**: A layered architecture reduces system complexity by providing an easy to understand structure. This increases interchangeability and maintainability by reducing coupling between system components. Figure 10 shows the implemented overall layering structure of the software. On the right-hand side, the different control loop levels and the sensor data flow are marked.

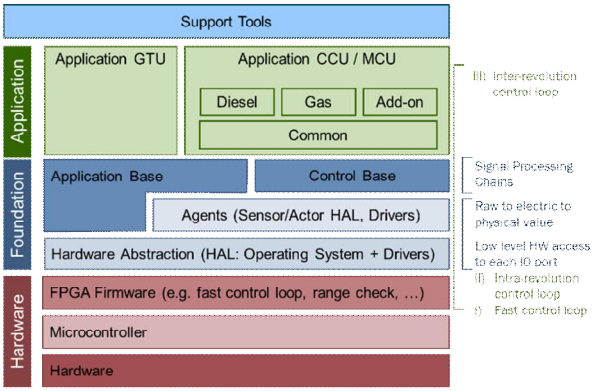

Figure 10. Layer Structure

3.3.1 Hardware and Microprocessor Layer

These various layers form the basis on which the entire software runs. They provide basic building blocks like CPU, RAM and flash memory that are needed to run software.

3.3.2 FPGA Layer

The FPGA provides blocks used by the software to communicate, read from sensors and to control actuators. It thus ensures that all the hard-real-time requirements can be fulfilled and provides the flexibility to serve the different control modules.

The FPGA is used, for instance, for the following tasks:

- **Fast Control Loop:** The fast control function for controlling a PWM output is fully implemented in the FPGA.
- **Intra-Revolution Control Loop:** This is supported by hardware blocks in the FPGA to ensure that other software blocks do not need to provide real-time accuracy.
- **Message Forwarding:** The System Bus uses Ethernet in a ring topology, for which purpose the FPGA bridges messages between modules, meaning an Ethernet switch is not needed, and also simplifying cabling and providing redundancy.

- **Safety Functions:** As a first approach, the FPGA enforces range checks.
- **Crank Angle Measurement:** This is one of the most important measurement functions of the system.

3.3.3 Hardware Abstraction Layer (HAL: Operating System and Drivers)

The HAL contains the operating system and all drivers that enable low level hardware access to input/output ports. For the FPGA, this provides access to the internal registers. The operating system contains all needed drivers for the SoC present on the hardware module.

The HAL and the Agents Layer above both deal with hardware blocks. The separation of components into two different layers enhances the software system testability.

Implementation of the HAL on a PC replaces hardware specific access methods with software emulators. This makes it possible to run the ECS Foundation on a normal host system for development, testing and simulation purposes.

Examples: CAN, Ethernet, etc.

3.3.4 Agents Layer (Sensor/Actuator HAL, Drivers)

The Agents Layer builds upon the HAL. The interface of the agents' layer to the upper layers provides physical values such as temperature readings.

For the sensors, drivers read a raw value from an input or output port via the HAL layer and convert it into a physical value. This conversion may be split into two steps:

- Conversion of a raw binary value from an analog-to-digital converter or digital sensor into an electrical value. To correctly map the raw value, this step may need sensor-specific tables from the sensor datasheet.
- Conversion of an electrical value from the step above into a physical value. To correctly convert the electrical value, this step needs sensor-specific tables, characteristic curves and/or calibration data. The data is set by the applications running on the ECS Foundation during start-up.

Examples are Analog Input Agent and Drive Agent.

3.3.5 Application Base Layer

The application base layer contains more generic components that facilitate routine tasks to unburden the application software. An existing software framework is used for the basic functions. System-specific components used by all modules belonging to this layer. Examples are lower functions that cover value validity and range checks or functions dealing with the redundant system bus e.g. functions that select the appropriate sensor value out of the redundantly available sensor values. Examples are software update, logging.

3.3.6 Control Base or Processing Blocks Layer

The processing blocks layer provides control function specific components. These components may be configured and parameterized. Control modules in the application layer synthesize their control functions by connecting various components from this layer. Important measurement functions like the crank angle measurement are at this layer. Examples are filters, FFT, crank angle access, etc.

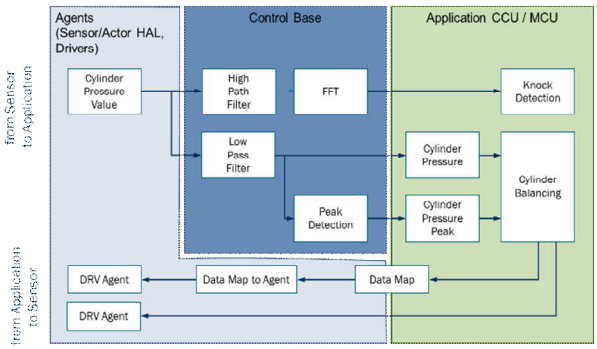

Figure 11: example of a sensor data processing chain

An example of a Sensor Data Processing Chain is shown in Figure 11. The processing starts with the cylinder pressure measurement as the input. The Agents Layer converts the raw binary sensor value read from an IO port into a physical value. The processing blocks layer in the control base preprocesses these physical values with signal processing blocks. In the example, the physical pressure value traverses several filter chains:

- High-pass filter with subsequent FFT
- Low-pass filter
- Peak detection based on low-pass filtering

Signal processing blocks pass their output values via ring buffers to the subsequent blocks. Each block puts its output values in a ring buffer. The size of a ring buffer is designed to be sufficiently large such that copying of values by subsequent blocks can be avoided. Several blocks can read

from a ring buffer. Each of these reading blocks has its own read pointer into the ring buffer. Examples are FFT, low-pass filter and peak detection results provided to the application layer.

3.4 Application Software

A basis for the application software has already been developed: extensive control algorithms are in operation of the entire X-engine family. These applications are, therefore, already realized modularly. This organization allows simplified maintenance and troubleshooting, as well as easy porting of applications to different platforms.

The control algorithms and state-machines are realized using the MATLAB/Simulink/Stateflow® toolchain, as shown by the example in Figure 12, and are auto-coded. The required effort for development of the new control system is thus reduced; significantly, there is also already a basis with proven behavior from existing deployment in the field.

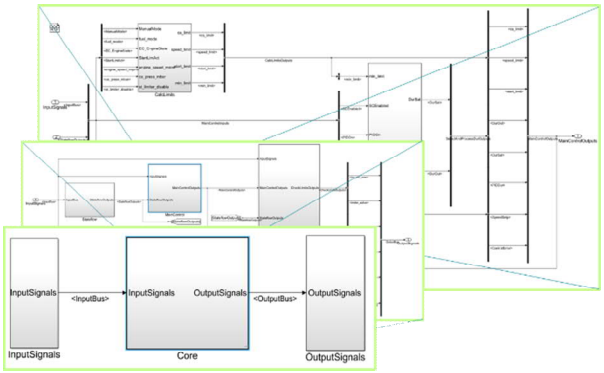

Figure 12. Simulink models of the speed control module – an example with three levels.

The first engine with the new WiCE system is the X92DF. This engine is equipped with the injection control unit (ICU), and the related control element had been modularized based on the WECS control algorithm and subsequently integrated to the X-engine control structure.

The migrated control functions of the engine can be clustered into main categories such as the systems governing the crank angle, fuel oil, gas injection, scavenge air, lubrication, servo oil, and air start process. Further functions are combined with common functionality, auxiliary functions, failure handling and monitoring algorithms.

Actuators to be controlled include the main fuel injectors, gas admission valves, exhaust valves, pilot fuel injectors, pilot start air valve, and cylinder lubrication valve. This is performed using measurements from sensors such as position feedback for fuel injection control, exhaust valve and gas admission valves, knock and cylinder pressure sensors, and several temperature and pressure sensors.

The control functions are executed on different HW modules; this is, where possible, implemented such that control functions are executed on the same module to which the input/output signals are connected. The cylinder specific control applications are executed on the CCUs, while common functions are dealt with by the MCUs. Information to be shared among cylinders is available via the Ethernet ring.

The process pattern follows the principle steps: read data (e.g. signals, sensor values), process data (computation), write data (set actuator, send message). These steps differ in terms of required calculation speed and support from the hardware. To accommodate these different properties, the processing chain is split into three control loop levels as already depicted in Figure 10:

- **Fast Control Loop (I)** has an update frequency of ~2 ...~15 kHz; ~ 500...65 us) and each loop directly controls one physical value. It is either implemented fully in the hardware or is supported by a hardware block. This type of loop is realized in the FPGA logic already described.
- The **Intra-Revolution Control Loop (II)** - within one revolution - is also a fast control loop. Again, this is realized in the FPGA logic and can be assisted by the Foundation Software. For instance, it can activate a solenoid (pull-in), hold and release it. The changes need to be applied exactly at a specific event or accurately after a specific time duration.
- The **Inter-Revolution Control Loop (III)** is a slower control loop (update frequency is every revolution max ~2.8 Hz -> ~350 ms) that adjusts values of the Intra-Revolution Control Loop. This loop is realized in the application software. For example, event and time duration for the Intra-Revolution Control Loop are set before an actuator gets active.

3.5 Software Integration and Testing

The WiCE system has many interactions between its various components and can be represented as complex interaction network. WiCE has been developed as an integrated software system with the aim of improving dependability and functionality through maintenance and evolution.

3.5.1 System Software Verification

The embedded software is written to control two-stroke engines. The embedded system itself is an information processing system that consists of both hardware and software components.

A major component of embedded systems is the software, which must be properly and adequately tested, especially concerning safety critical domains. Due to the high complexity, defects in the control system can cause critical situations and delays which can create huge costs. Therefore, a combination of different analysis methods is applied with the aim of verifying and validating components in the electronic control system. Several testing methods are used according to the level of development shown in Figure 13.

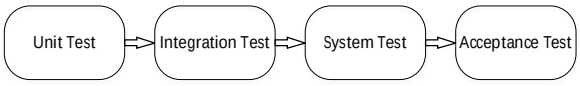

Figure 13. SW Test sequence

3.5.2 Application Software Test on Computer

Verification and validation are basic activities to ensure a properly working system. Functional verification involves, as a first step in development, a model testing approach to verify whether each function of the system is working according to the requirements and specifications.

3.5.3 Model Tests

A model test procedure (Figure 14) is employed to check each model according to requirements. By using smart tools like the “Design Verifier” of Mathworks, errors can be identified at a very early stage in the development process. A further advantage of model checking is that the verification can be carried out automatically and test cases can be reused.

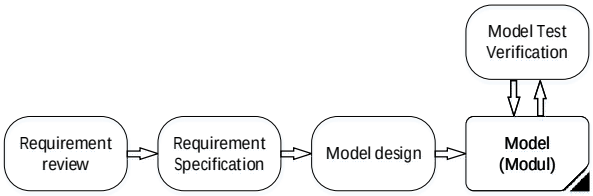

Figure 14. Model testing

These tools enable functional, baseline, equivalence, and back-to-back testing, including software-in-the-loop (SIL), processor-in-the-loop (PIL), and real-time hardware-in-the-loop (HIL) methods. It allows pass and fail criteria to be applied that include absolute and tolerance limits, logical checks, and temporal conditions. Test scripts help automate and customize test execution, as well as to store test cases and their results. Generating reports, archiving and reviewing test results, rerunning failed tests, and debugging components are key testing activities.

3.5.4 Software Integration

Integration is the combination of subroutines and software modules with other software components to develop an application or enhance the functionality of an existing one. Often requiring a large degree of source code modification, developers may spend a large amount of their time performing software integration. The model-based approach with a defined interface to the system software helps mitigate the above-mentioned situation.

3.5.5 Static Analysis

Static analysis is done with the generated code by means of modern tools. Without involving dynamic execution of the software under testing possible defects can be detected at an early stage. By an automatic “walk through” of the source code it detects code that violates set rules. This ensures that proper coding standards and conventions are used to construct the program.

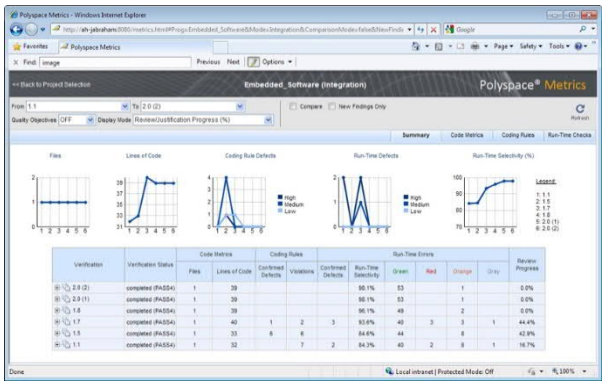

Figure 15. Test report of “White-box testing”

The tool (Figure 15) identifies run-time errors, concurrency issues, security vulnerabilities and other defects in the code. Using static analysis, including semantic analysis, allows the analysis of data flows and other behavior. By highlighting defects as soon as they are detected, bugs can be fixed very early in the development process.

Automatic compliance checks the code against coding rule standards such as MISRA C [5] and custom naming conventions. It generates reports consisting of bugs found, code-rule violations, and code quality metrics, including cyclometric complexity. The tool supports automatically

generated code and results can be traced back to level of the models.

3.6 Hardware-in-the-loop Testing

Hardware-in-the-loop simulation (HIL) is an integral component in the electronic development process for testing control functions. HIL simulation involves testing electronic control units (ECUs) in a closed loop with components that are simulated in real time in order to test them intensively. Hardware-in-the-Loop simulation is a method used in the product development cycle in which one or more real components interact with components that are simulated in real time (dynamic models).

The part of the system that is not simulated can consist of real devices. Nowadays, however, the term is mainly used to refer to a real system that comprises one or more CCUs, controllers, or some mechatronic components for which a virtual environment is simulated. The interactions between the real and the simulated subsystems impose heavy real-time demands on the latter. The WinGD HIL test platform is shown in Figure 16.

Figure 16. Hardware-in-the-Loop (HIL) test rig. Visible are the CCUs, left side, and two MCPs and several load cards for actuators on the right-front side.

The simulated subsystem has to perform several actions: reading in all signals, executing the algorithm, and outputting the results. The result is a closed loop which includes the real controller and the simulated plant.

The major aims of HIL simulation are to enhance product quality, to reduce complexity and to lower development costs. This last aim is achieved by moving function tests and diagnostics tests from tests drives or test bench experiments to the HIL laboratory. The result is that the number of expensive tests on a real engine prototype and time spent at the test bench are considerably reduced. HIL tests can be reproduced as often as required.

Nowadays, it is common practice to completely automatically run, evaluate, and document tests overnight or on weekends. This allows the operators to concentrate on assessing tests, implementing new tests or test scripts, and adjusting tests that may have failed. Automation provides far greater test coverage than manual tests, and this enhances quality and maturity.

Frequently, the HIL simulators are also used to perform tests that would not be possible at all on a real system, since these would be dangerous to test on real engine.

Therefore, by HIL simulation it possible to give an ECU an environment that is sufficiently realistic to allow the control system to detect any inconsistencies in the electrical design and the dynamic behavior of the plant. In this situation, any tests that would typically be performed with the engine (or on a test bench) can also be performed on the HIL simulator. This is the basis for intentionally inducing error situations. Diagnostic functions (with HW faults, functional faults or communication errors), fail-safe levels and emergency operation can be tested in this way.

3.7 Tests on Real Engine

For sure, the final test of the control system must happen on the real application. Figure 17 gives an impression of the first WiCE installation on the test engine in Winterthur.

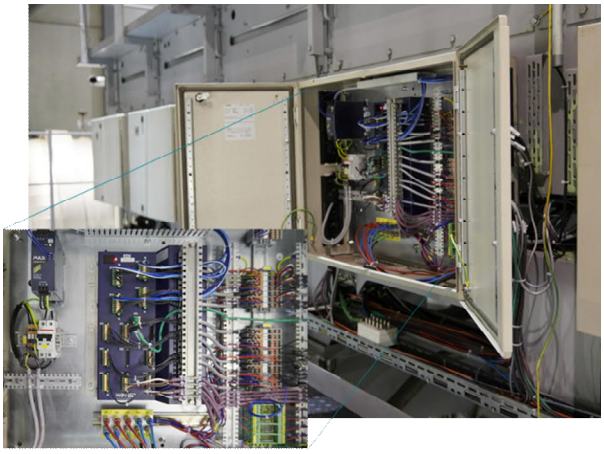

Figure 17. Control system on test engine RTX-6

At the time of writing, written, the tests on the test engine RTX-6 have been started and first results are available. All tests described beforehand will be repeated on this higher integration level. In addition, the values measured by the control system will be crosschecked by measurements with special measurement equipment. These tests will be continued on several engines and installations until all requirements and needs for safety and security are ensured and fulfilled.

4 SOFTWARE ENVIRONMENT

4.1 Configuration system

SW Configuration (SCM) is the field of management focused on establishing and maintaining the consistency of its system or product performance and its functional and physical attributes. The system and product requirements, design, and operational information documentation must be maintained throughout the lifetime of a system.

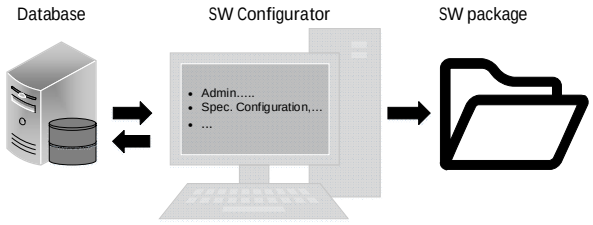

Figure 18. SW Configurator

The change control of the configuration process manages the security features and quality assurance throughout the life cycle of an information system to hardware, software, firmware, documentation, test, test fixtures, and test documentation. Processes and tools realize this control. Delivery processes and tools for maintaining the base configuration and supporting deployment of the individual SW are thereby established, Figure 18.

4.2 Software Deployment

Several activities are subsumed under expression deployment:

- **Release**: Activities followed by the completed SW development process (D-Message), including operations to prepare a system for assembly and transfer to the customer site (e.g. determining parameter values). It must determine the resources required to operate at the customer site (e.g. personnel, documents, tools, training, etc.).
- **Installation** (activation): Installation and activation of the configured SW package (SW-Download and restart of the control system).
- **Deactivation**: Refers to shutting down any executing components of a system (SW-Update).
- **Adaptation**: Modification of a previously-installed package (e.g. change of parameter).
- **Update**: Replacement of an earlier version, all or part of a software system with a newer release (SW-Update).
- **Version tracking:** This helps find and install updates to installed packages (how to handle feedback, feedback tracking).

An Agile SW delivery process, maintenance of the base configuration including SW Configuration Tools is the key for success in this area.

4.3 Commissioning tool

Commissioning can serve many purposes. One of the most important of these is reducing the risk of unplanned outages and downtime. A proper commissioning ensures that the engine is in the best state to function as intended.

Commissioning is a quality-oriented process for achieving, verifying, and documenting that the performance of the facility and assembly meets defined objectives and criteria. Commissioning

was developed in response to the recognition that the normal design, construction, and systems start-up process could frequently not ensure a trouble-free outcome. Commissioning engineers are equipped with sophisticated tools to get rapid access to key information on a control system.

The flexView2 tool, shown in Figure 19, described below, supports SW down- and uploads. It is easy to use and practical, supporting commissioning engineers carry out their important role quickly and efficiently. Due to the fact that flexView is already known, the handling with flexView2 is not challenging.

Figure 19. flexView2: Basic page

flexView2 is the commissioning and trouble-shooting tool for engines equipped with WiCE, and allows software packages to be to down- and uploaded. It allows parameters to be accessed as seen in Figure 20 and specific functions to be selected for monitoring engine operation. It is designed to provide extended trending, monitoring and testing facilities. Standard functions for engine operation are accessible via the user interface of the propulsion control system or via manual control panels.

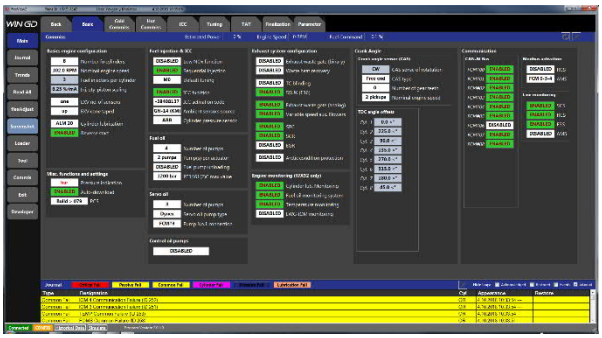

Figure 20. flexView2: Commissioning page for parameter input

The tool, which is connected via Ethernet cable contains powerful functions to enable WinGD experts on board to evaluate the condition of the engine, and provides a wide range of tools to support troubleshooting operations, Figure 21.

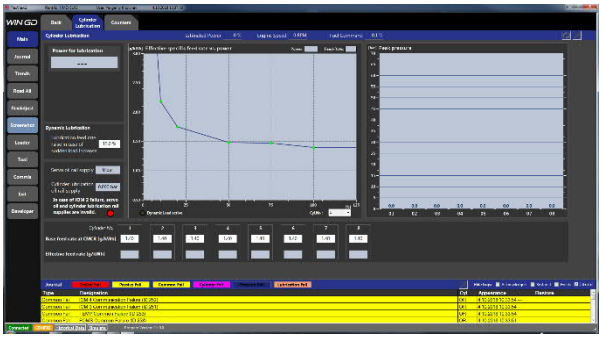

Figure 21. flexView2: commissioning page with trends

5 SUMMARY

WiCE is a scalable platform for machinery control, consisting of several types of modular devices (units), which can be put together to build differently sized arrangements. In the case of the main engine control functionality, a distributed system is built that introduces variety of new concepts and standards compared to the current systems. Some examples:

- State-of-the-art software and hardware architectures, allowing easy scalability and lifecycle management.
- Ethernet ring for data communication and Crank angle signal distribution; enabling faster data transmission.
- New bus system topology for internal and external communication, contributing to efficient and lean installation.
- Hardware hierarchy with redundant main controllers, gateway units and one control unit per cylinder, clustering signals and functions.
- A firewall, separating essential engine control functionality from peripheral systems, minimizing the risk of system breach
- New HW design with powerful logic components and flexible I/O types; fulfilling the demands for the years to come.
- New Development environment and module interface for software development and configuration.
- New commissioning and monitoring tools built-up on the feedback from the field making the system intuitive for the operator.

With WiCE WinGD can offer a state-of-the-art electronic control system that meets the demanding requirements for control of marine two-stroke engines. The WiCE platform meets existing demands such as complex and autonomous control with enhanced diagnostics, remote monitoring including measures against cyber-attacks. It is designed to be easily integrated into vessels' existing electronic infrastructure. Experienced engine operators will recognize the upcoming system as beneficial as well as the promising possibilities concerning further expansion.

6 DEFINITIONS, ACRONYMS, ABBREVIATIONS

ALM: Alarm and Monitoring System

ASW: Application Software

CCU: Cylinder Control Unit

CPU: Central Processing Unit

ECR: Engine Control Room

EMC: Electromagnetic Compatibility

FFT: Fast Fourier Transformation

FPGA: Field Programmable Gate Array

FW: Firmware

GTU: Gateway Unit

HIL: Hardware in the Loop

HW: Hardware

ICU: Injection Control Unit

MAC: Media Access Control

MCP: Manual Control Panel

MCU: Main Control Unit

PCS: Propulsion Control System

PIL: Processor in the Loop

RAM: Random-Access Memory

SIL: Software in the Loop

SOC: System on (a) Chip

SW: Software

UNIC: Wärtsilä Unified Controls

WECS: Wärtsilä Engine Control System

WiCE: WinGD integrated Control Electronics

WiDE: WinGD integrated Digital Expert system

WinGD: Winterthur Gas & Diesel Ltd

7 ACKNOWLDEDGEMENTS

The project is executed as part of the innovation program of and financed by CSSC (China State Shipbuilding Cooperation).

8 REFERENCES AND BIBLIOGRAPHY

[1] Revised MARPOL Annex VI, Regulations for Prevention of Air Pollution from Ships - and NOx Technical Code 2008, *International Maritime Organisation*, amended 2014

[2] Future Ship Powering Options – Exploring alternative methods of ship propulsion, *Royal Academy of Engineering*, London, UK, July 2013.

[3] Global Marine Technology Trend 2030 – Autonomous Systems, *Lloyds Register Group Ltd, QinetiQ and University of Southampton*, UK, 2017

[4] Low Carbon Shipping towards 2050, *DNV GL AS*, Hovik, Norway, 2017

[5] MISRA (Motor Industry Software Reliability Association), *MISRA-C – set of software development guidelines*, https://www.misra.org.uk/

[6] Cartalemi, C. Meier, M., A real time comprehensive analysis of the main engine and ship data, *CIMAC Congress 2019*, Vancouver, Paper no. 349

[7] Schneiter, D., Nylund, I., Alder, R., Printz, P., Goranov, S., Ott, M., 12X92DF - The development of the largest ever Otto cycle engine, *CIMAC Congress 2019*, Vancouver, Paper no. 425

This paper has been presented and published at the 29th CIMAC World Congress 2019 in Vancouver,Canada. The CIMAC Congress is held every three years, each time in a different member country. The Congress program centres around the presentation of Technical Papers on engine research and development, application engineering on the original equipment side and engine operation and maintenance on the end-user side. The themes of the 2019 event included Digitalization & Connectivity for different applications, System Integration, Electrification & Hybridization, Emission Reduction Technologies, Low Carbon Combustion including Global Sulphur Cap 2020, Case Studies from Operators, Product Development of Gas and Diesel Engines, Components & Tribology, Turbochargers, Controls & Automation, Fuels & Lubricants as well as Basic Research & Advanced Engineering.