Contents

[**1. INTRODUCTION** 2](#_Toc463958762)

[1.1 Overview of the Company 2](#_Toc463958763)

[1.2 Thesis Problem. 3](#_Toc463958764)

[1.3 Thesis outline. 3](#_Toc463958765)

[**2. Background.** 4](#_Toc463958766)

[2.1 State Of Art. 4](#_Toc463958767)

[2.1.1 Vendor independent type: 4](#_Toc463958768)

[2.1.2 Model based: 7](#_Toc463958769)

[Bibliography 10](#_Toc463958770)

# **1. INTRODUCTION**

## 1.1 Overview of the Company

Endress+Hauser is a global leader in measurement instrumentation, services and solutions for industrial process engineering. The company provides process solutions for flow, level, pressure, analytics, temperature, recording and digital communications, and optimizing processes in terms of economic efficiency, safety and environmental impact. Our customers come from various industries, including chemical, food & beverage, life sciences, power & energy, primaries & metal, oil & gas and water & wastewater. The company’s five production centers with headquarters in Germany and Switzerland focus on knowhow in research and development, product management as well as logistics. At these sites they also manufacture core components for our worldwide production.

Endress+Hauser Maulburg is one of the leading producers of level and pressure instrumentation. The company employs more than 2,000 associates world-wide. Headquartered in Maulburg, near to the French and Swiss border, Endress+Hauser has also sites in Kassel and Stahnsdorf. Associated Product Centers in Greenwood (USA), Suzhou (China), Yamanashi (Japan), Aurangabad (India) and Itatiba (Brazil) are responsible for customized final assembly and calibration of measuring instruments.

* Dedication to quality: Quality and continuous improvement is a priority at Endress+Hauser. We strive for exemplary quality in our products and services - regardless from which Endress+Hauser facility they come from.
* A solution for every task: We provide customers with hundreds of thousands of measuring points every year: virtually every device is unique, selected from a catalog with 2,000 different products and one billion available design variants.
* Innovation from day one: Founded in 1953 Endress+Hauser was a true pioneer in electronic measuring technology. Since then we have continued to convince our customers with trailblazing products and solutions.
* Minimum integration effort: To ensure that our instruments can be integrated into control and asset management systems with minimum effort, we test them in a multi-vendor environment and ensure they have the appropriate certification before putting them on the market.

## 1.2 Thesis Problem.

In order for the new products to reach the customers as early as possible every aspects in the project from planning to certification has its own importance. During the development of an electronic board, an important phase is the commissioning of the board. Though this phase, the hardware (HW) developer have to verify if the board is correctly developed, if the layout is correct and if there are some failures in the production. Furthermore, in a later step, some EMC tests can be done with the board. Generally, there is at least one microcontroller on the board. This means that, to test the board during the commissioning, this microcontroller has to be programmed. The program controls all interfaces and peripherals, such that each peripheral component on the board can be tested. However, some HW developers do not have software development skills. This means they have to wait on software developers to develop test software.

So, the idea is to develop a user Friendly PC tool which replaces the software developer. This tool will generate test software for the target microcontroller according to some inputs from the HW developer. This test software will control the defined peripherals and interfaces that the HW developer wants to test/verify.

## 1.3 Thesis outline.

This thesis flow is structured in the following manner

**Chapter 2 deals with requirements and state of art.**

**Chapter 3 deals with implementation.**

**Chapter 4 deals with testing and optimization.**

**Chapter5 concludes the thesis with future work.**

# **2. Background.**

Before we go to the requirements let us understand the process of producing a new electronic boards. The first step to any product development is the buildup a strategy by placing the problem discussion among a group of product designers. When the strategy is built the next steps would be:

1. Component Selection: To, select the various IC’s, sensors, displays, connectors and other electronic devices based on the desired function. A detailed block diagram is helpful at this step.
2. Schematic: schematic diagram shows how the components are connected together. This is the heart of designing electronics.
3. Printed Circuit Board (PCB) Design: when the schematic is created it is handed over to a PCB designer. This is where you will created the design for the physical board which holds and connects all the components according to the schematics.
4. PCB Prototypes: A board prototype is prepared by first developing the bare circuit board and then having all the electronic components soldered onto the board.
5. Evaluate, Debug and Repeat: The first version of the prototype is evaluated and almost it is never the final. The issues need to be debugged and fixed for the next time and this step is repeated several times. This can be a difficult step to forecast in terms of cost and time.

So, to reduce the time in the evaluation, debugging step. The idea is to develop an automatic code generator which helps the electronic engineer to commission the section of the board by just selecting the components and its functionality. The output is then crosschecked for the designed functionality.

## 2.1 State Of Art.

There are some tools which serve for the similar problem by creating a standardization to lower the complexity

### 2.1.1 Vendor independent type:

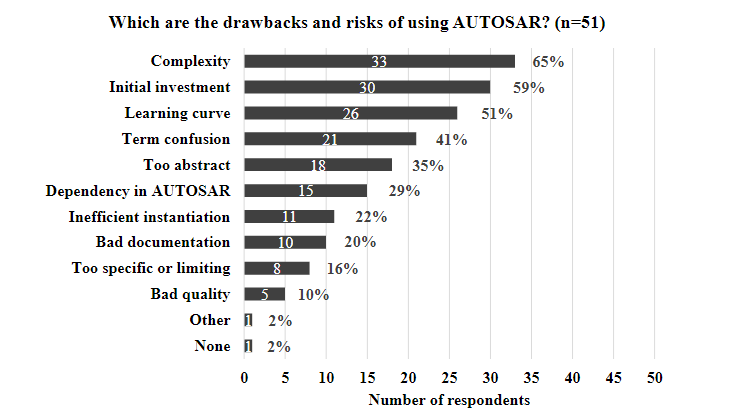
1. **AUTOSAR** [1]: AUTOSAR (AUTomotive Open System ARchitecture) is open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers. The AUTOSAR-standard enables the use of a component based software design model for the design of a vehicular system. The design model uses application software components which are linked through an abstract component, named the virtual function bus.

The application software components are the smallest pieces of application software that still have certain functionality. The software of an application can then be composed by using different application software-components. Standardized interfaces for all the application software components necessary to build the different automotive applications are specified in the **AUTOSAR**-standards. By only defining the interfaces, there is still freedom in the way of obtaining the functionality.

The virtual function bus connects the different software components in the design model. This abstract component interconnects the different application software components and handles the information exchange between them. The virtual function bus is the conceptualization of all hardware and system services offered by the vehicular system. This makes it possible for the designers to focus on the application instead of the infrastructure software.

By using the virtual function bus, the application software components do not need to know with which other application software components they communicate. The software components give their output to the virtual function bus, which guides the information to the input ports of the software components that need that information. This is possible due to the standardized interfaces of the software components which specify the input and output ports as well as the format of data exchange.

This approach makes it possible to validate the interaction of all components and interfaces before software implementation. This is also a fast way to make changes in the system design and check whether the system will still function.

The most mentioned drawbacks and risk of using the AUTOSAR in a survey 

1. **CMSIS**: The **Cortex Microcontroller Software Interface Standard** (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series. The CMSIS enables consistent and simple software interfaces to the processor and the peripherals, simplifying software re-use, reducing the learning curve for microcontroller developers, and reducing the time to market for new devices.

The CMSIS is defined in close cooperation with various silicon and software vendors and provides a common approach to interface to peripherals, real-time operating systems, and middleware components. The CMSIS is intended to enable the combination of software components from multiple middleware vendors.

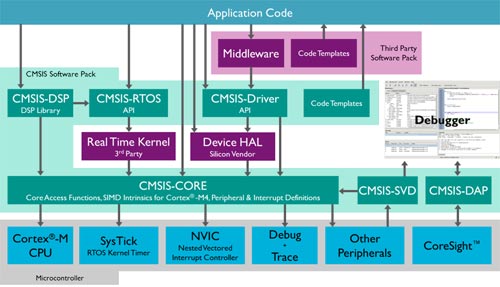
The CMSIS components are:

**CMSIS-CORE**: API for the Cortex-M processor core and peripherals. It provides at standardized interface for Cortex-M0, Cortex-M3, Cortex-M4, SC000, and SC300. Included are also SIMD intrinsic functions for Cortex-M4 SIMD instructions.

**CMSIS-DSP**: DSP Library Collection with over 60 Functions for various data types: fix-point (fractional q7, q15, q31) and single precision floating-point (32-bit). The library is available for Cortex-M0, Cortex-M3, and Cortex-M4. The Cortex-M4 implementation is optimized for the SIMD instruction set.

**CMSIS-RTOS API**: Common API for Real-Time operating systems. It provides a standardized programming interface that is portable to many RTOS and enables therefore software templates, middleware, libraries, and other components that can work acrosss supported the RTOS systems.

**CMSIS-SVD**: System View Description for Peripherals. Describes the peripherals of a device in an XML file and can be used to create peripheral awareness in debuggers or header files with peripheral register and interrupt definitions.



**CMSIS Structure**

Motivation

CMSIS has been created to help the industry in standardization. It is not a huge software layer that introduces overhead and does not define standard peripherals. The silicon industry can therefore support the wide variations of Cortex-M processor-based devices with this common standard. In detail the benefits of the CMSIS are:

* Consistent software interfaces improve the software portability and re-usability. Generic software libraries can interface with device libraries from various silicon vendors.
* Reduces the learning curve, development costs, and time-to-market. Developers can write software quicker through an easy to use and standardized software interface.
* Provides a compiler independent layer that allows using different compilers. CMSIS is supported by all mainstream compilers (ARMCC, IAR, and GNU).
* Enhances program debugging with peripheral information for debuggers .

### Model based:

1. **Targetlink:** Software for automatic code generation, based on a subset of Simulink/State flow models, produced by dSPACE GmbH. TargetLink requires an existing MATLAB/Simulink model to work on. TargetLink generates both ANSI-C and production code optimized for specific processors. It also supports the generation of AUTOSAR-compliant code for software components for the automotive sector. The management of all relevant information for code generation takes place in a central data container, called the Data Dictionary.

Testing of the generated code is implemented in Simulink, which is also used for the specification of the underlying simulation models. TargetLink supports three simulation modes to test the generated code:

* Model-in-the-loop simulation (MIL): this mode allows the model design to be checked. An MIL simulation is also known as a floating-point simulation, since the variables are typically floating-point variables.
* Software-in-the-loop (SIL): the simulation is based on the execution of generated code, which runs on a PC system. The variables are typically plain or fixed point numbers.
* Processor-in-the-loop (PIL): in a PIL simulation, the generated code runs on the target hardware or on an evaluation board. So-called real-time frames are included, making it possible to transfer the simulation results as well as memory consumption and runtime information to the PC.

1. **Embedded Coder:** Generates readable, compact, and fast C and C++ code for use on embedded processors, on-target rapid prototyping boards, and microprocessors used in mass production. Embedded Coder enables additional MATLAB Coder™ and Simulink Coder™ configuration options and advanced optimizations for fine-grain control of the generated code’s functions, files, and data. These optimizations improve code efficiency and facilitate integration with legacy code, data types, and calibration parameters used in production. You can incorporate a third-party development environment into the build process to produce an executable for turnkey deployment on your embedded system.

From the Simulink Model Explorer, you can:

* Generate code for your Simulink models and subsystems
* Select an Embedded Coder target
* Configure the target for code generation
* Create, load, and reuse multiple configuration sets

### 2.1.3 Vendor specific tools:

1. **ATMEL:** Atmel Start helps you getting started with Atmel® microcontroller development. It allows you to select MCU, configure software components, drivers, middleware, and example projects to tailor your embedded application in a usable and optimized manner. Once you are done you can download the generated code project and open it in Atmel Studio or another third party development tool.

With Atmel Start you can:

* Get help selecting MCU based on both software and hardware requirements
* Find and develop examples for your board
* Configure drivers, middleware, and example projects
* Get help with setting up a valid PINMUX layout
* Configure system clock settings

1. **STMicroelectronics:** STM32CubeMX is part of STMicroelectronics STMCube original initiative to ease developer’s life by reducing development efforts, time and cost. STM32Cube covers STM32 portfolio.

STM32Cube includes the STM32CubeMX which is a graphical software configuration tool that allows generating C initialization code using graphical wizards.

It also embeds a comprehensive software platform, delivered per series (such as STM32CubeF4 for STM32F4 series). This platform includes the STM32Cube HAL (an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio), plus a consistent set of middleware components (RTOS, USB, TCP/IP and graphics). All embedded software utilities come with a full set of examples.

STM32CubeMX is a graphical tool that allows configuring STM32 microcontrollers very easily and generating the corresponding initialization C code through a step-by-step process.

Step one consists in selecting the STMicroelectronics STM32 microcontroller that matches the required set of peripherals.

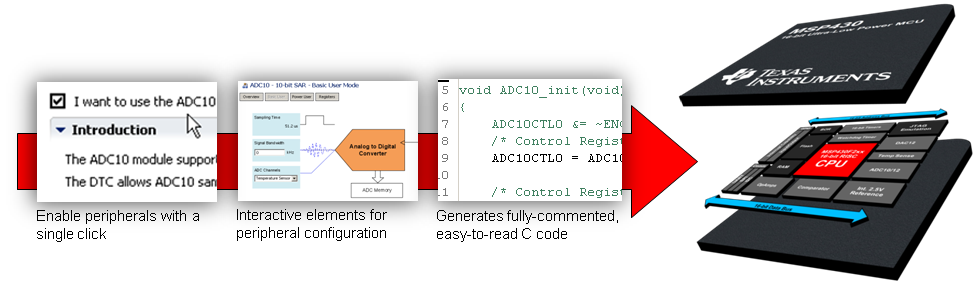
The user must then configure each required embedded software thanks to a pinout-conflict solver, a clock-tree setting helper, a power-consumption calculator, and an utility performing MCU peripheral configuration (GPIO, USART, ..) and middleware stacks (USB, TCP/IP, ...).

Finally, the user launches the generation of the initialization C code based on the selected configuration. This code is ready to be used within several development environments. The user code is kept at the next code generation.

**Key Features**

* Intuitive STM32 microcontroller selection
* Microcontroller graphical configuration:
  + Pinout with automatic conflict resolution
  + Clock tree with dynamic validation of configuration
  + Peripherals and middleware functional modes and initialization with dynamic validation of parameter constraints
  + Power sequence with estimate of consumption results
* C code project generation covering STM32 microcontroller initialization compliant with IAR, Keil and GCC compilers.

1. **Texas Instruments (TI):**
2. **HALCoGen**: A tool from TI provides a graphical user interface that allows the user to configure peripherals, interrupts, clocks, and other microcontroller parameters. Once the device is configured, the user can generate peripheral initialization and driver code, which can be imported into CCS, IAR Workbench, or Keil.
3. **GRACE:** Grace is a tool that allows a MSP430 developer to generate the peripherals setup code within minutes.



Once you're done enabling and configuring the various peripherals required for the application, Grace will generate fully-commented, easy-to-read C-code at build time. Unlike other options that are available, which output cryptic assembly language, Grace generates C-code that is fully modular and is automatically inserted into the active project in Code Composer Studio.

This code can be debugged and downloaded to the MSP430 device just like traditionally written code.

2.1.1 User interface:

The user interface should be easy to use should not have complicated manipulations so that even without any prior knowledge on the microcontroller programming an user should be able to perform his required test.

Should have almost all the possibilities are frequently tested.

The flow structure should be easily understood.

Adding and removing of the user interface to be tested should be easy.

2.1.2 Developer.

Should have a great overview of the list of peripherals and their usage limits.

He should mind the

2.1.3 Maintenance.

Here the term maintenance refers to the process of the developing and updating the tool.

For the developers succeeding it should be comprehensive and easy to start developing.

Developing the tool here refers to adding a new microcontroller to the list and its features and also updating the peripherals for the existing controllers.

# Bibliography

|  |  |
| --- | --- |
| [1] | S. Maradana, "automotive basics," [Online]. Available: https://automotivetechis.wordpress.com/autosar-concepts/. |