

INTERNSHIP REPORT

Thesis by
Syed Muhammad Saim (2180551)

In Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Engineering - ELECTRONICS



HOCHSCHULE
HAMM-LIPPSTADT

HOCHSCHULE HAMM-LIPPSTADT

University Examiner: **Prof. Dr. Stefan Henkler**

!!!!

Defended Company Supervisor: **Steffen Gugenhan**

ACKNOWLEDGEMENTS

During my Internship at STAR Electronics GmbH Co. KG, several people helped me much in a unique way. Firstly, I would like to thanks my first supervisor Akshay Kulkarni who guided me through the company and gave me essential knowledge related to his department, Gateway Testing. He also supported me in understanding of the Automotive Bus Systems and Communication Protocols and how they work. In addition, I am also grateful to Javad Mazyar who helped in setting up the environment for CAPL Generator Software. Moreover, I am deeply indebted to Andreas Merath who guided me through Ranorex Testing and gave me essential knowledge related to Automation and Quality Assurance department. Also, I would like to extend my deepest gratitude to Selim Cetin who was also very helpful and knows much regarding this department with whom I worked with. Furthermore, I would like to thanks Konstantin Klaus who helped me in testing the hardware in the Development Department while working at Black-Box Testing. Eventually, a special thanks is to Steffen Gugenhan who placed me in different departments of the company and hence, I now have a whole picture of what is the main goal of the company and how they are pursuing it.

ABSTRACT

This internship report provides an overview of the duties carried out during the required internship at STAR ELECTRONICS GmbH Co. KG in Göppingen. It is a division of Star Cooperation, and its main goals are to create the appropriate tools and offer solutions for complicated car electronics. Original Equipment Manufacturers (OEMs), Tier 1, Tier 2, and Automotive Supplier Companies all employ their products. The internship report begins with a thorough corporate overview.

The Report begins with the introduction of the company and brief overview of the tools that I have used. In addition, concrete overview have also been described for the Automotive Bus Communication Systems including CAN, CAN-FD, FlexRay and SOME/IP in Chapter 3 Page 7. Moreover, the report talks about the tasks that I have completed during my whole tenure from Chapter 4 Page 11 onwards. Describing Task from every department begins with a brief Motivation to get an idea what is the purpose of that specific area. The description goes on while explaining the procedure and mentioning my contributions and hence, drawing conclusions.

A thorough conclusion is drawn towards the end of the internship report, and a thorough discussion of the skills acquired during the internship is included. Additionally, the pros and cons of the internship are examined, as well as how it will affect my career in the long run.

TABLE OF CONTENTS

Acknowledgements	iii
Abstract	iv
Table of Contents	v
Chapter I: Introduction of Company	1
1.1 Softwares	2
1.2 Hardwares	2
Chapter II: Tools Used	4
2.1 Jenkins	4
2.2 Tortise SVN	4
2.3 Intel Quartus Prime	5
2.4 Ranorex	5
Chapter III: Training - Automotive Bus Systems and Communication Protocols	7
3.1 FlexConfig RBS	7
3.2 Scalable service-Orientend MiddlewarE over IP (SOME/IP)	8
3.3 Controller Area Network (CAN)	9
3.4 Controller Area Network Flexible Data-Rate (CAN-FD)	9
3.5 FlexRay	9
Chapter IV: TASK 1: Gateway Testing	11
4.1 Motivation	11
4.2 CAPL GUI	12
Chapter V: TASK 2: Black-Box Testing - Field Programmable Gate Array (FPGA) Testing	16
5.1 Motivation	16
5.2 Flashing	17
5.3 Complex Programmable Logic Device (CPLD)	18
5.4 Flex Tinys	19
5.5 Cables	20
5.6 My Contributions	21
Chapter VI: TASK 3: Regression Testing	23
6.1 Motivation	23
6.2 Test Setup	24
6.3 My Contributions	25
Chapter VII: Internship Evaluation	28
7.1 Academic Relevance	28
7.2 Skills Learned	28
7.3 Pros And Cons of the Internship	29
Appendix A: Consent Form	30

Bibliography

Chapter 1

INTRODUCTION OF COMPANY

STAR Electronics GmbH Co. KG is a German service company with its headquarters in Böblingen, which is also represented at other locations in Germany, North America, Africa and Asia. The company supports other companies in the planning, design and implementation of projects, the core business takes place in the automotive sector [13].

Star Electronics GmbH Co. KG is part of the STAR Cooperation and belongs to the Electronics and Automotive Electronics division. In the area of Electronics we develop, design and manufacture the appropriate tools for complex vehicle electronics. The experts at STAR Electronics accompany customer companies in their various projects from automotive E/E-systems and bus systems to measurement technology and vehicle conversion [13].



Figure 1.1: Fields of Activity of STAR GmbH Co. KG [13]

As a specialist in automotive electronics, STAR Electronics provides consulting services for complex concepts. STAR Electronics supports companies with the right hardware and software. In addition, standard and individual special tools for precise measurement and simulation technology, sophisticated control technology as well as sustainable energy supply and efficient energy management [13].

1.1 Softwares

FlexConfig RBS

For all concerns relating to automotive bus systems, there is just one place to go: FlexConfig RBS. You may quickly simulate, view, edit, and record bus data using this program. With the aid of hardware, it can also be utilized to generate Remaining Bus Simulations. There are many different bus systems and network description formats available. The other bus simulations are created and executed on a variety of hardware. Among the hardware models supported are FlexDevice-S, FlexDevice-L, FlexDevice-L2, FlexCard PXIe3, and FlexCard PCIe3. FlexRay, CAN high-speed, CAN low-speed, CAN FD, LIN, SENT, Ethernet (100BASE-TX/1000BASE-T), and Automotive Ethernet (100/1000BASE-T1) are a few of the bus systems that FlexConfig RBS supports. The remaining bus simulators can likewise be designed, built, and maintained with great success using this technology [8].

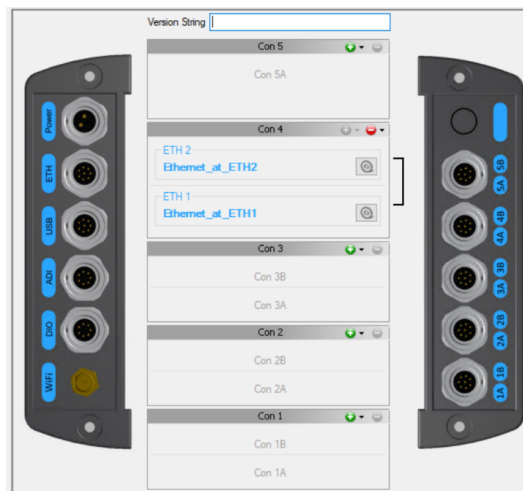


Figure 1.2: FlexConfig RBS Software [8]

1.2 Hardwares

FlexDevice-S



Figure 1.3: FlexDevice-S [11]

The FlexDevice-S versatile bus control unit can be utilized in a variety of bus-related applications thanks to its single configurable interface and compatibility with both current and future bus/network systems. This device is popular among

embedded software developers because it enables developing and testing even the most sophisticated gateway programs and prototype features simpler. Developers may efficiently create and maintain their bus configurations and ECU environments with the FlexConfig software series. Up to eight persons can fit on the FlexDevice-S [11].

FlexDevice-L



Figure 1.4: FlexDevice-L [9]

The FlexDevice-L is a high-performance instrument in the FlexDevice product series, which is utilized and regarded globally in the development of automotive electronics by numerous automakers and component suppliers. This version of FlexDevice is more sophisticated and contains a ton of new capabilities. The ARM Cortex A9 Dual Core (800 MHz) with 1 GByte DDR3 RAM, an integrated WiFi and Bluetooth module, up to 7 1000BASE-T1 bus interfaces, and 5 bus connectors that may be flexible allocated using pluggable transceivers are a few noteworthy features [9].

FlexDevice-L²



Figure 1.5: FlexDevice-L² [10]

FlexDevice-L2, the newest member of the FlexDevice product line, is an enhanced version of FlexDevice-L. Two ARM Cortex A9 Dual Core processors running at 800 MHz with 2 GB of DDR3 RAM each, five bus connectors that may be assigned in any way, up to 20 CAN-HS/CAN-FD bus interfaces, and both internal and external Micro SD card ports are a few of the extra features of the device [10].

Chapter 2

TOOLS USED

2.1 Jenkins

Jenkins is a tool for building, delivering, and executing automated tests that is open-source and free. Jenkins' most important function in this Use Case was continuous integration. Continuous Integration (CI) is the process of regularly integrating code updates from various developers into a single project. After a code commit, the software is instantly tested. Every change necessitates the writing of fresh code and its testing. The build is then tested for deployment if the test is successful. The code is then sent to production if the deployment is successful. Continuous integration/deployment is the process of regularly producing, testing, committing, and deploying code. Depending on the test results, Jenkins reports are generated and the appropriate action is done [5].

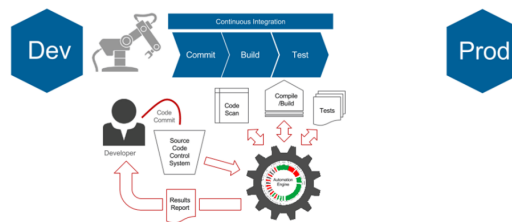


Figure 2.1: Jenkins Workflow [5]

2.2 Tortoise SVN

TortoiseSVN is a Windows program for revision control, version control, and source control. It is built on Apache Subversion (SVN)®, and TortoiseSVN offers a user interface for Subversion that is attractive and simple to use [16].

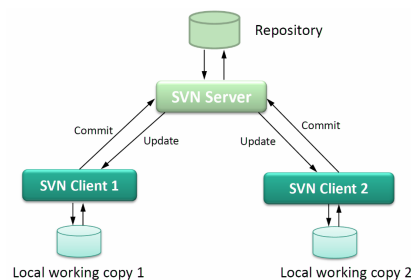


Figure 2.2: Tortoise SVN Workflow [16]

2.3 Intel Quartus Prime

For the most advanced Intel® Agilex™, Intel® Stratix® 10, Intel® Arria® 10, and Intel® Cyclone® 10 GX FPGA and SoC designs, this software offers a full design environment. Easy design entry, quick design processing, simple device programming, and integration with other industry-standard EDA tools are all supported by the Intel® Quartus® Prime software GUI. You can easily concentrate on your design rather than the design tool thanks to the user interface. The modular Compiler simplifies the design of FPGAs and guarantees the best performance with the least amount of work [3].

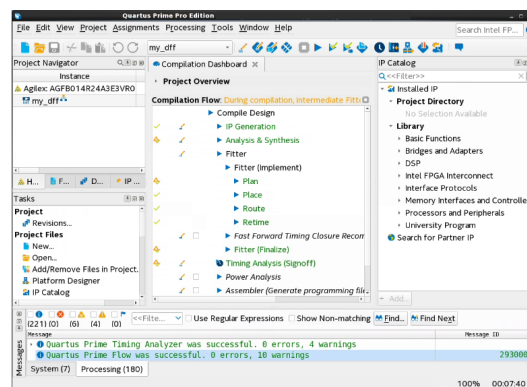


Figure 2.3: Intel Quartus Prime Software [3]

2.4 Ranorex

Ranorex Studio is a GUI test automation framework provided by Ranorex GmbH, a software development company. The framework is used for testing desktop, web-based and mobile applications. Using Ranorex Studio's complete automation IDE, tests are automated using CSharp or VB.NET. Reliable automated test cases can be created by the developer using Ranorex record and replay feature even without programming [4].

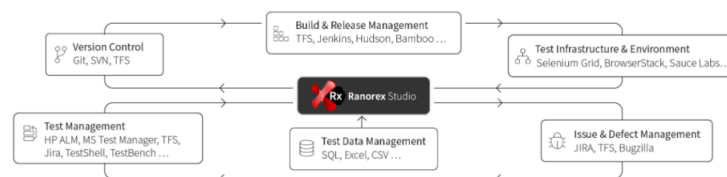


Figure 2.4: Ranorex Workflow [4]

Test automation, like team collaboration, is essential for developers and testers. Ranorex test automation project offers different types of usage, each designed for

specific skills in a cross-functional team. Developers and technological testers use Ranorex core automation framework to develop program codes for flexible automation elements. Test results and project progress can be reviewed using XML-based test reports [4].

Chapter 3

TRAINING - AUTOMOTIVE BUS SYSTEMS AND COMMUNICATION PROTOCOLS

3.1 FlexConfig RBS

The FlexConfig-RBS training is divided into presentations and practical parts. The biggest part of the presentations is related to the use cases of the software mentioned as well as to own tips and tricks to be able to fix configuration errors. The practical parts are planned to create trial projects. Under the use cases of the software are understood and brought close in practice [8].

FlexConfig RBS ensures the following use cases:

Residual Bus Simulation

Messages from one or more ECUs can be simulated [8].

Transparent Gateway

The Transparent Gateway is inserted from the shared bus or network and forwards all bus communication. The intervention in the communication is enabled by Transparent Gateway [8].

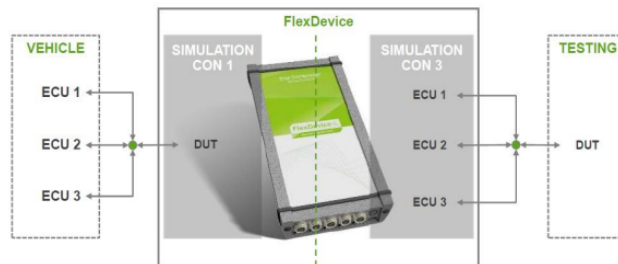


Figure 3.1: Transparent Gateway Structure [8].

FlexConfig Gateway

FlexConfig Gateway can be used to route data between different bus systems and allows scaling and conversion of database elements [8].

Bus Complementation

Complementing and enabling the various bus and network communication in the vehicle is achieved by several Bus supplements (e.g. Can, LIN Master/Slave or ETH-Link) [8].

Manipulation

Manipulations are interventions in the parameterization of a send message or interventions in the send behavior or interventions in the transmission behavior [8].

Filter

Only required data is sent. Other data are filtered out [8].

Visualization

Data (e.g. signals, data types) can be visualized on different bus systems networks with different scaling. Measurement and calibration systems, test benches and analog measurement modules are and measurement results for complex analyses are visualized [8].

3.2 Scalable service-Orientend MiddlewarE over IP (SOME/IP)

SOME/IP is an automotive/embedded communication protocol which supports remote procedure calls, event notifications and the underlying serialization/wire format [14].

SOME/IP provides service oriented communication over a network. It is based on service definitions that list the functionality that the service provides. A service can consist of combinations of zero or multiple events, methods and fields. Events provide data that are sent cyclically or on change from the provider to the sub-scriber. Methods provide the possibility to the subscriber to issue remote procedure calls which are executed on provider side. Fields are combinations of one or more of the following three

- a notifier which sends data on change from the provider to the subscribers
- a getter which can be called by the subscriber to explicitly query the provider for the value
- a setter which can be called by the subscriber when it wants to change the value on provider side [14]

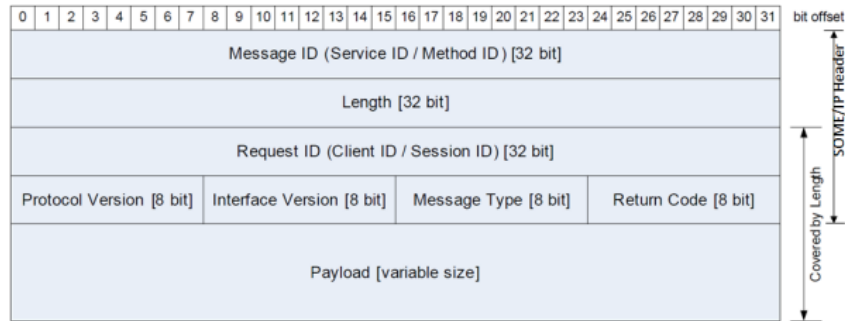


Figure 3.2: SOME/IP [14]

3.3 Controller Area Network (CAN)

CAN is defined as a message-oriented, multi-master, serial asynchronous communication bus that uses the ISO/OSI reference model's two bottom levels. CAN is largely used in the automotive industry for dependable data exchange between electronic control units (ECUs). Automobiles, building automation, elevators, and lightning control systems are just a few examples of typical applications. CAN has a number of drawbacks, including overhead bits, the use of two wires, and latency time [6].

3.4 Controller Area Network Flexible Data-Rate (CAN-FD)

CAN FD is upgraded to the original CAN protocol that aims to increase data transfer rates up to five times those of a CAN frame by switching to a faster bit rate and using a different frame type. The necessity to bridge the performance gap between CAN HS (1Mbit/s) and FlexRay (10Mbit/s) drove its development. The data length has also been raised from 8 bytes to a maximum of 64 bytes. [7]

3.5 FlexRay

The FlexRay protocol can transport data at a rate of 10 megabits per second over each of its two channels, whereas the CAN standard can only send data at 1 megabit per second. This translates to a data rate of 20 Megabits per second, which is twenty times faster than a CAN-based system. FlexRay systems are appropriate as the cornerstone of a network backbone even where CAN is already in use due to their high data rate. The FlexRay protocol can handle a variety of bus topologies. These include point-to-point links, passive stars, linear passive buses, active star networks, cascaded active stars, hybrid topologies, and dual channel topologies [12].

	CAN	CAN FD	FlexRay v3.0.1
Message IDs	11 or 29 bit		11 bit
Data Payload	8 bytes	64 bytes	254 bytes
Bit rate	1 Mbit/s	AP: 1Mbit/s, DP: 8Mbit/s	2.5, 5 or 10 Mbit/s
CRC	15 bit	17 bit (16 bytes) 21 bit (64 bytes)	15 bit header CRC 24 bit trailer CRC
No. Channels	1	1	2 (A and B)
Network Access	CSMA-CD NDBA or Time Triggered	CSMA-CD NDBA or Time Triggered	Time Triggered
Chip Support	Most micros	Bosch, Freescale, NXP, ST	Many 16 / 32-bit micros
AUTOSAR Support	Yes	v4.1.1 CAN FD 8 bytes v4.2.1 CAN FD 64 bytes	Yes

Figure 3.3: CAN, CAN-FD and FlexRay Comparison [12]

Chapter 4

TASK 1: GATEWAY TESTING

4.1 Motivation

A high-quality final product is produced in large part through testing. When working with Gateway testing, standard operating procedure was adhered to. Figure 4.1 shows how the work flow begins with the test setup, where a test setup is created using a Vector Box, which is a device created for automotive industry by Vector GmbH, and a FlexDevice. Depending on the use scenario, the configuration of the devices can vary. Step 1 is followed by creating a gateway project in the FlexConfig RBS and flashing the appropriate binary file onto the FlexDevice (s19 File). Moving forward, the FlexConfig RBS exports CSV, and the Communication Access Programming Language (CAPL) generator generates the CAPL code. The Vector CANoe, which is a software for automotive industry created by Vector GmbH, runs the resulting CAPL code, and HTML reports are produced.

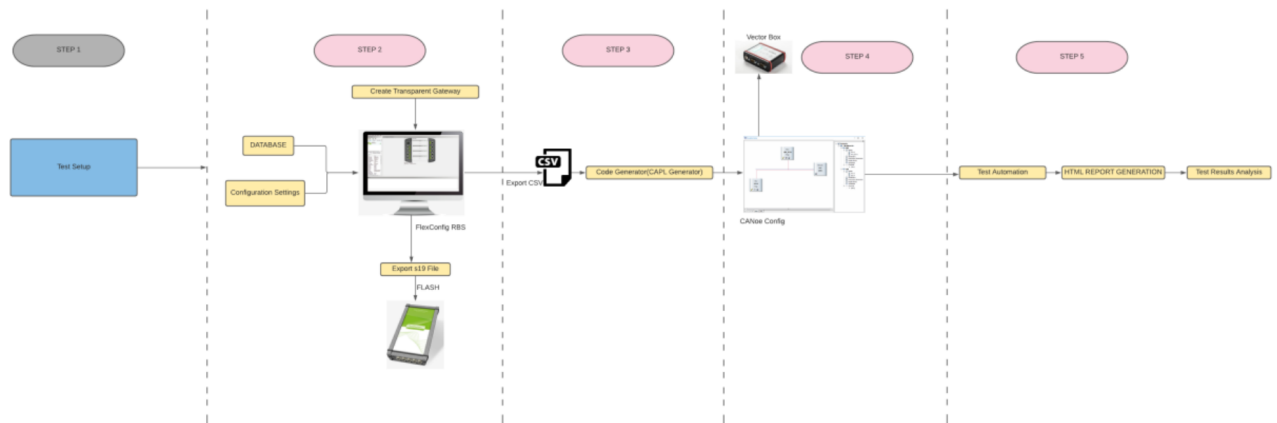


Figure 4.1: Gateway Testing workflow

The results of each test must next be verified by carefully going over the generated reports. Failures are carefully investigated, and the underlying source of the problem is identified and fixed. My task was to focus on the third step where I had been given a csv file and I had to verify that the actual code is being generated using CAPL Generator GUI.

4.2 CAPL GUI

Knowing how to use the CAPL Generator was essential because it was a key part of the Gateway testing. Let me simply outline the general concept before delving into depth about my contributions to the CAPL generator. A CAPL generator was used to build the CAPL code that would be executed on the Vector CANoe. The code for the CAPL generator was also written in OOP CSharp. A CAPL generator, to put it simply, is a piece of code that analyzes a csv file and produces a short bit of CAPL code based on the parameters listed in the csv. The generation of the CAPL code came after the construction of the FlexConfig RBS project.

The Graphical User Interface of the CAPL generator is depicted in Figure 4.2. There are two boxes to check. The timely delivery of the frames was verified by cyclic tests. Additionally, choosing the test timing duration was an option. Simply drag and drop the csv file onto the GUI to use it as an input parameter.

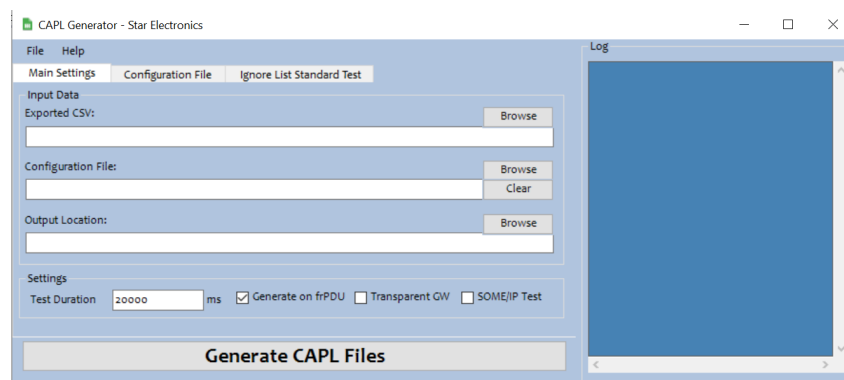


Figure 4.2: CAPL Generator Software

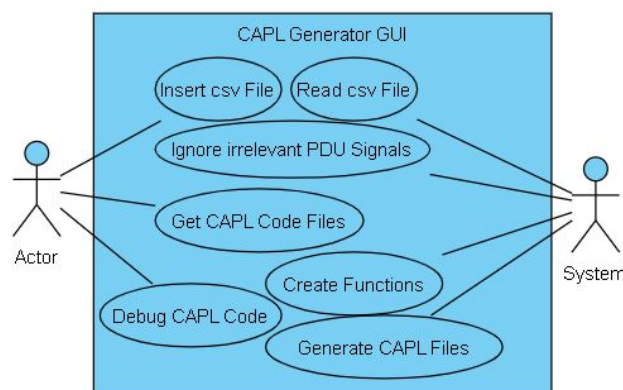


Figure 4.3: CAPL Generator Use-Case Diagram

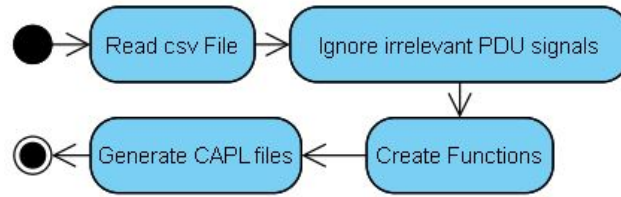


Figure 4.4: CAPL Generator Activity Diagram

Commonly used .NET classes in the CAPL generator's CSharp code include String builders and T4 Templates. T4 Templates (Text Template Transformation Toolkit) can be thought of as a combination of text blocks and control logic that can build a text file. String Builders is used anytime several string operations are necessary.

My Contribution 1

During my tenure in this department, the task was to identify why some of the paramount signals from the csv were ignored by the GUI and the generated files from this GUI were incomplete. I debugged the code on Visual Studio and found the errors. In addition, I also changed some of the programs in a compact form. Previously, it was written with from, where and select keywords. I changed the code into one single line by using .Where() method which can be found inside LinQ library of C.

My Contribution 2

The information under the heading "Getter Response" in the csv file was being ignored by the PreFilter method of the GUI. This particular method in the C code was removed and changings in the GetDistinctECUs() method was added. In this method the variable methodConsumerECU was taking ECU from the heading "Receive" in a csv file which was changed to "Consume" after understanding how SOME/IP works. A check was also placed with a variable named matchConfig of type bool. This ensured that if the condition "item.Equals(ecuList)" becomes true, item would be added to the allEcuClean variable and matchConfig variable turned to true which assured the completion of the program.

```

1 bool matchConfig == false;
2 foreach (var item in singleEcu.Split(','))
3 {
4     foreach (var ecuList in supplementalConfiguration.TargetECUs)
5     {
6         if (item.Equals(ecuList))
7         {
8             allEcuClean.Add(item);
9             matchConfig = true;
10            break;
11        }
12    }
13    if (matchConfig == true) break;
14 }

```

Listing 4.1: CSharp

My Contribution 3

Another task came up in which the structure of the generated file had to be designed on switch statements. This had to be done because new information was going to be added into the databases. The information that was present in the functions inside generated files needed to be grouped. To fulfill this task, I applied multiple grouping concepts from the LinQ library of CSharp. I used one variable `dataSet` in my `GenTest_2` method which has information read from the csv in a list form. I had to group this information at least 3 times. Using the `GroupBy` and `Select` method with multiple keys, the results were converted to lists by using the `ToList` method which gave flexibility to loop through the values while considering variable `groupByTargetServiceItem` as lists. For outputting the data, an object had been created “output” of the class `StringBuilder` and added each output by using the `.Append()` method. To print the results, string concatenation and string interpolation had been implemented. This method, which I wrote, comprises 78 lines. The first `foreach` loop loops through the `groupByTargetServiceItem` where it considers elements before the `.Select()` method. The second `foreach` loop loops through the elements after the `.Select()` method. Lastly, the third `foreach` loop loops through `Target_Reciver_ECUs` that are splitted and stored `targetEcu` with the help of a conditional statement that checks whether values are available or not.

```

1 var groupByTargetServiceItem = dataSet.GroupBy(d => new {
2     d.Target_Service_Item,
3     d.Source_Service_Major_Version,
4     d.Source_Service_Instance,
5     d.Source_Service,
6     d.Target_Service_Major_Version,
7     d.Target_Service_Instance,
8     d.Target_Sender_ECUs,
9     d.Target_Service
10 }).Select(a => new{
11     targetServiceItem      = a.Key.Target_Service_Item,
12     targetSignalName      = a.Select(b => (b.
13     Target_Signal_Name, b.Target_Receiver_ECUs, b.Index)),
14     sourceServiceMajorVersion = a.Key.Source_Service_Major_Version
15     ,
16     sourceServiceInstance    = a.Key.Source_Service_Instance,
17     sourceservice = a.Key.Source_Service.Split('_'),
18     targetServiceMajorVersion = a.Key.Target_Service_Major_Version
19     ,
20     targetServiceInstance    = a.Key.target_Service_Instance,
21     sourceEcu                = a.Key.Source_Sender_ECUs,
22     targetService            = a.Key.Target_Service.Split('_')
23 }).ToList();

```

Listing 4.2: CSharp

Chapter 5

TASK 2: BLACK-BOX TESTING - FIELD PROGRAMMABLE GATE ARRAY (FPGA) TESTING

5.1 Motivation

Black-box testing is a technique for evaluating software that looks only at the functionality of an application without looking at its internal components or operations. Practically every level of software testing, including unit, integration, system, and acceptance testing, may be conducted using this test methodology. Testing that is based on specifications is another name for it [1].

It is not necessary to have detailed understanding of the application's code, internal structure, or general programming knowledge. Although the tester is aware of what the software is intended to perform, they are unaware of how it actually accomplishes this. For example, the tester may be aware that a specific input leads to a specific, deterministic outcome but may not be aware of how the software generates the output in the first place [1].

The steps which were followed while testing are explained in the further sections which includes Flashing, Complex Programmable Logic Device (CPLD), FlexTiny and Cables. The procedure is also explained in the Use Case Diagram.

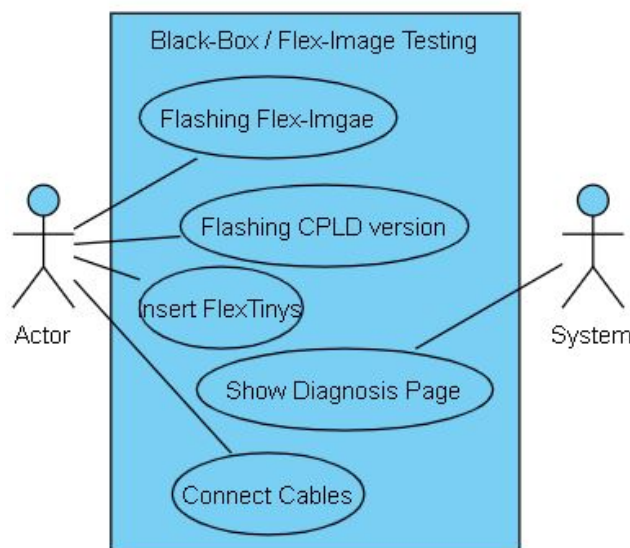


Figure 5.1: FPGA-Image Use Case Diagram

5.2 Flashing

The department of FPGA Testing mostly belongs to hardware testing. The FPGA images software, which are developed by the embedded team, are flashed on the Flex Device L, L² and S, and it was tested whether the developed software was complying with the hardware or not. Each Flex Device and Flex Card had unique FPGA images software and these images were flashed via FlexTFTP, which is a known developed TFTP tool to flash fpga images on Flex Devices and Flex Cards easily, software by drag and drop the file.

Flex-Image, a template, is a configuration file for the FPGA module, which is installed on the Flex Device. The configuration file defines which communication controllers and therefore which buses are available. The project setup in FlexConfig RBS has to be decided on such a template.

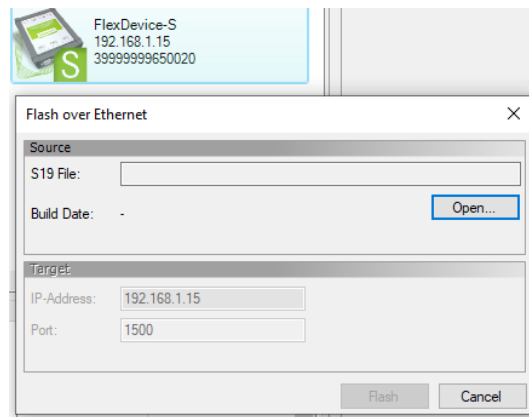


Figure 5.2: Flashing

The next step was to flash the binary file (.s19) file which is the project file. Project file is based on some specific instructions on how the device should be set up for a specific FPGA image. Both of these files required thirty to forty seconds to be flashed and the flashing is confirmed as the light flashes on the device. This file first has to be generated by the software Flex Config RBS via build option.

After flashing, the data can be seen on the diagnosis page. The diagnosis page has all of the device information that includes various hardware and their software versions. In addition, it also shows communication between various channels.

STAR COOPERATION® FlexDevice-L 2SoC 2nd Gen System Diagnosis

Your Partners in Excellence

Hardware & Versions | Flash | Threads | Debug | Log | Self Check | Ethernet | Statistics | Analyzing | Logging | Services | Time Sync | Trace | Disks | WiFi | Support

Self check encountered no problems

General

Application State:	Running
Device Name:	FlexDevice-L 2SoC 2nd Gen <input type="text"/> save
Serial:	30010201620030
Hardware-Version:	5.2.0.0
Hardware-Type:	FlexDevice-L 2SoC 2nd Gen (8)
Feature Licenses:	RBS, Gateway, Control, Analyzing, Logging <input type="button" value="Update Licenses"/>
FL3X Config Licenses:	No licenses
Up Time:	00:00:27h
Boot Time:	1<- FPGA/QSPI 176ms -> <- Basics 18ms -> <- Communications 0ms -> <- Application 10ms -> <- Analyzer 0ms -> <- FR Sync 45ms ->
	<------ 204ms -----> <------ 249ms ----->
RBS Version:	FlexConfig RBS 5.5.0.4059
RBS Time Stamp:	637890781645668488 (25.05.2022 12:22:44h)
Compile Time:	May 25 2022 - 12:22:49
Project Name:	Siw-Aufbau0_2_UT
Manipulations State:	Not Active
Internal SD-Card:	<input checked="" type="checkbox"/> Phison 64GB SN:39PH21686343
External SD-Card:	<input checked="" type="checkbox"/> SanDisk 393GB SN:35D1507470932
Extension Board:	WiFi + µSD-Card Slot (0V01-20 K044 / 2021 / SN:30010195830027 / Firmware: 1.0.4.0)
CPU Temperature:	24°C
Supply Voltage:	11.78V
RTC Buffer Voltage:	2.33V
Local RTC Time:	2022-05-27 10:17:15 set time
Software Sleep:	<input type="checkbox"/>
WakeUp Source:	Power On
IP / Netmask / Gateway:	192.168.1.15 / 255.255.255.0 / 192.168.1.1 <input type="button" value="save"/>

Firmware

Power Management CPLD:	43 (0x0028)
System Control CPLD:	4145 (0x1031)
Preloader:	1.5.0.0
Factory CPU:	1.0.2.0
Factory FPGA:	1.1.3.4
Distribution CPU:	255.255.255.255
Revisions:	05(25982) CAI(26020 - 21.04.2022-14:20:53,69) SC(12187 - 21.04.2022-14:22:13,68)
RBS API:	1.7.0.0
FPGA1:	
Loaded Image Version:	0.0.0.6
Loaded Image Type:	0x1186
Needed Image Version by RBS:	0.0.0.0
Needed Image Type by RBS:	0x1186
Init Done / Config Done:	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
FPGA2:	
Loaded Image Version:	0.0.0.5
Loaded Image Type:	0x1090
Needed Image Version by RBS:	0.0.0.0
Needed Image Type by RBS:	0x1090
Wait Time:	90719µs
Init Done / Config Done:	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>

Figure 5.3: Diagnosis Page

5.3 Complex Programmable Logic Device (CPLD)

The first tab under the heading Hardware and Versions shows all of the hardware components and their respective versions. One such hardware is the Complex Programmable Logic Device (CPLD). It has two main versions that had to be flashed depending on the FPGA image software. To flash a version of CPLD, Intel Quartus Prime Software had to be used together with the Joint test Action Group (JTAG). The JTAG device is connected to one of the slots inside Flex Device L² which is then connected with CPLD, FPGA and CFI. CFI is the device where the FPGA images (software) are stored and CPLD is used to read those images and hence, program the FPGA. All of this happens inside the Flex Device. The hardware setup diagram is also shown within the Quartus Prime Software.

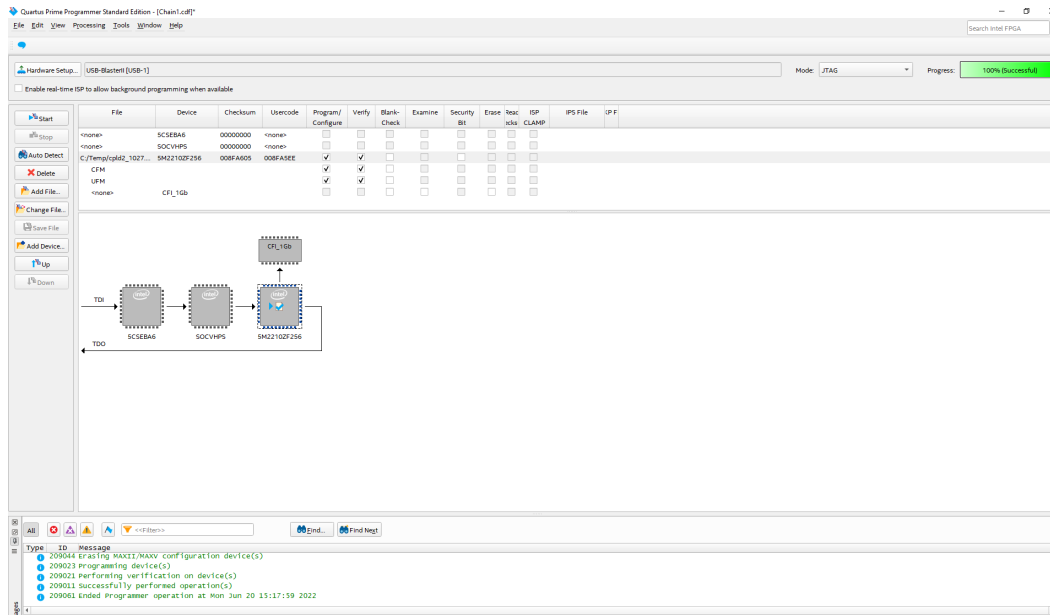


Figure 5.4: Intel Quartus Prime with connected hardware.

First of all, the hardware setup button is clicked, and hardware is selected as USB-BlasterII and frequency is set to 16000000 Hz. The device is detected automatically by clicking the Auto Detect button. In addition, boxes are checked to make sure that everything is verified. Eventually, success status in the progress bar is shown in the software that the version of the CPLD has been uploaded successfully.

5.4 Flex Tinys

If the page is scrolled down, FlexTinys could be seen missing. These are some unique small sized systems on board chips that had to be inserted on the empty slots of the Flex Device L². These represent various bus systems such as CAN-FD, CAN-HS, LIN, FlexRay and Ethernet. Moreover, every Tiny has a unique code to distinguish it from others. The binary file (.s19) that was flashed before, gave various configurations that showed which of the Tinys had to be inserted. Based on a required test, the five combinations of these Tinys are inserted into the five empty slots on the Flex Device.

FlexTinys					
Slot:	1	2	3	4	5
Tiny Type:	Empty	Empty	Empty	Empty	Empty
Physical Layer:					
Sleep Support:	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>
Wake Up Support:	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>
Configurable Termination:	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>	A <input type="checkbox"/> B <input type="checkbox"/>
Production Date:	Empty	Empty	Empty	Empty	Empty
Hardware Version:					
Product Number:					
Batch Number:					
Additional Info:					
Expected by RBS:	Missing	Missing	Missing	Missing	Missing

Figure 5.5: FlexTinys Information

While placing the Tinys on the device, utmost precaution must be considered that one has to be grounded to avoid any static charge. The device was placed on a special table mat that was connected to a switch to make the area grounded. In addition, I also had to wear a wristband which was connected to that mat. Safety shoes were also a necessity.

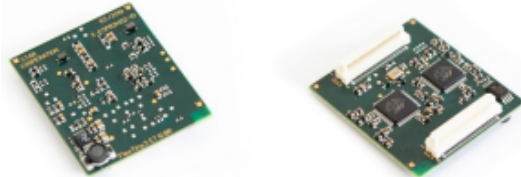


Figure 5.6: FlexTinys [2]

5.5 Cables

Once the Tinys had been inserted, special cables, which were designed and developed for Flex devices and Flex Cards, had to be connected into the slots called connectors. Our Flex Device is considered as Device Under Test (DUT). Cables represent various bus systems and each bus system had unique cables. Based on the information in the Project file and Tinys connected to the device, the cables are connected respectively.

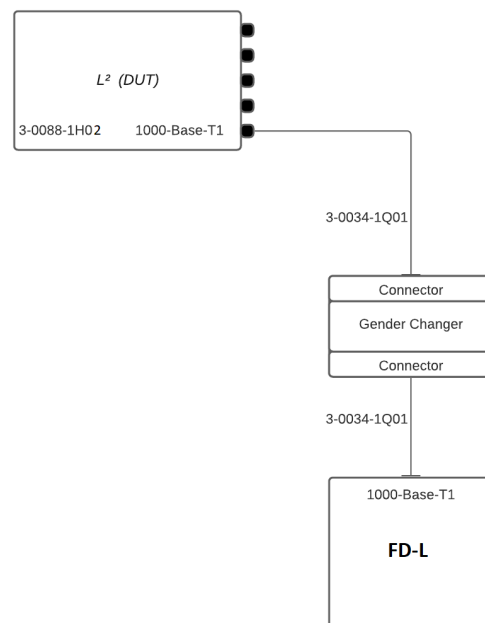


Figure 5.7: Cables

Once the correct cables had been connected and the results were shown on the diagnostic page under the heading Log that the cables are passing. The results

showed failing if the cables are not connected properly or there is any error in the generated FPGA image which had to be informed to the department.

Every bus channel has a receive and transmit and there are also numbers inside it. These numbers being shown are the frames that have been transmitted. The rate of these frames may vary and sometimes it takes too much time to reach 1000 frames after which every bus communication shows passing. To configure the standard rate of these frames a functionality has to be adjusted in FlexConfig RBS Software.

```
[INFO] Loaded Image FPGA1: Type(109B) Version(107)
[INFO] Loaded Image FPGA0: Type(118E) Version(F)
[TEST] ETH1 (Mux)Length Test: PASSED (146)
[TEST] ETH1 Receive: PASSED (7256) Transmit: PASSED (7207)
[TEST] ETH2 (Mux)Length Test: PASSED (146)
[TEST] ETH2 Receive: PASSED (7256) Transmit: PASSED (7206)
[TEST] ETH3 (Mux)Length Test: PASSED (145)
[TEST] ETH3 Receive: PASSED (7253) Transmit: PASSED (7203)
[TEST] ETH4 (Mux)Length Test: PASSED (145)
[TEST] ETH4 Receive: PASSED (7256) Transmit: PASSED (7206)
[TEST] CAN-FD 1, Receive: PASSED (7513) Transmit: PASSED (7304)
[TEST] CAN-FD 2, Receive: PASSED (7513) Transmit: PASSED (7304)
[TEST] CAN-FD 3, Receive: PASSED (7513) Transmit: PASSED (7304)
[TEST] CAN-FD 4, Receive: PASSED (7513) Transmit: PASSED (7304)
[TEST] FR1A Receive: PASSED (3662) Transmit: PASSED (7288)
[TEST] FR1A Offset Control: NOT CONFIGURED (0)
[TEST] FR1B Receive: PASSED (3662) Transmit: PASSED (7288)
[TEST] FR1B Offset Control: NOT CONFIGURED (0)
[TEST] FR2A Receive: PASSED (3645) Transmit: PASSED (7256)
[TEST] FR2A Offset Control: FAILED, also check CC Count (6761)
[TEST] FR2B Receive: PASSED (3645) Transmit: PASSED (7256)
[TEST] FR2B Offset Control: PASSED (7241)
```

Figure 5.8: Project Results

Eventually, the diagnostic page is saved and committed into the repositories so that other colleagues of the department can also investigate this.

5.6 My Contributions

I did this procedure to test many FPGA images and test cases in order to ensure that the developed FPGA images were correct, functional and ready to be deployed. I used to check that the right cables were being used for the right Bus Communication System and The Tinys were connected properly according to the project description. In addition, I used to verify that all of the signals, routings and channels are not giving any errors under the headings "Self Check", "Statistics" and "Test" of the diagnosis page. Finally I used to save the results and notified the department after which these FPGA images were added into the software FlexConfig RBS.

During my tenure in this particular department, the new version of the FlexConfig RBS Software 5.5 was released, and I was mesmerized that I became a part of that release by testing and ensuring correct functionality.

Chapter 6

TASK 3: REGRESSION TESTING

6.1 Motivation

Regression testing is a technique used in software testing to verify that an application still performs as expected after any modifications, updates, or improvements to the code. The general stability and efficiency of the current features are ensured by regression testing. Regression testing is used whenever a new change is made to the code to ensure that the system continues to function properly even after each update [15].

Regression testing is frequently used in the following situations:

- An existing feature gains a new stipulation.
- There is a new function or feature added
- To address bugs, the codebase has been fixed.
- Performance is increased by optimizing the source code.
- patch updates are made
- Configuration alterations

An essential component of software development processes is test automation. Similar to this, automated regression testing is regarded as a crucial component. Product teams can obtain more detailed feedback and react immediately with a speedy regression testing procedure. Regression testing finds new vulnerabilities early in the deployment cycle, saving firms the expense and work of fixing the accumulated flaws. An apparently little change can occasionally have a cascading effect on the product's essential capabilities. Because of this, developers and testers must not permit any modification—no matter how small—that falls outside of their sphere of influence [15].

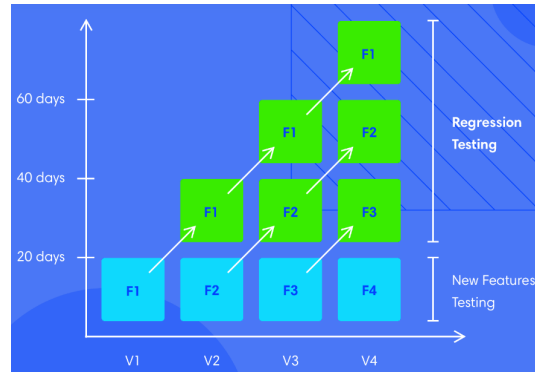


Figure 6.1: Regression Testing Workflow [15]

One such technique is using Ranorex software which will be explained in the further sections.

6.2 Test Setup

Figures 6.2 gives a general overview of Ranorex software that how the test is set up for FlexConfig RBS, which is Application Under Test (AUT). The green folders are classified as test suites whereas the folders inside these suites are known as smart folders where the recordings are placed. Recordings consist of series of actions and events that directs the flow of testing as shown in Figure.

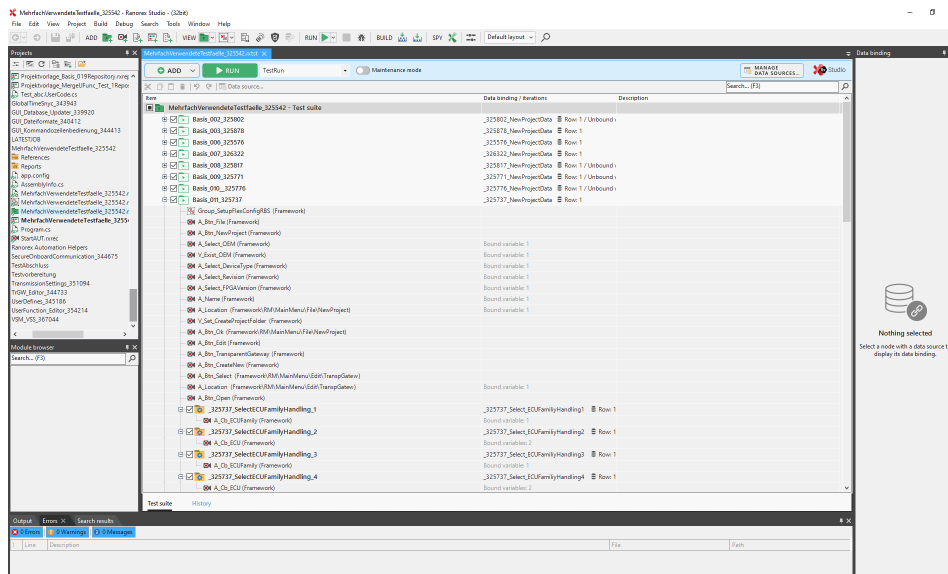


Figure 6.2: Ranorex Software with added Testsuites

After this set up has been established, the software is allowed to run through all of the recordings and shows the whole simulation where it checks every single feature

of the AUT. All of the tests were run the whole night and the reports of were stored in XML.

6.3 My Contributions

I had to look into the reports and identify the problems and alter the test procedures. One such way is to disable any specific recording which is not relevant and causing the problem. For instance, “A_Btn_Info_All_OEMs_OK (Framework)” is disabled as it is shown in italic in figure 6.3.

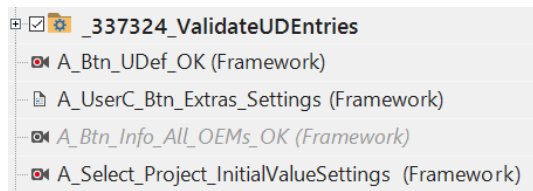


Figure 6.3: Disabling Button

Another way is to add actions and events through mouse and keyboard. This is done by clicking the recording button and record the action that were performed using peripherals.

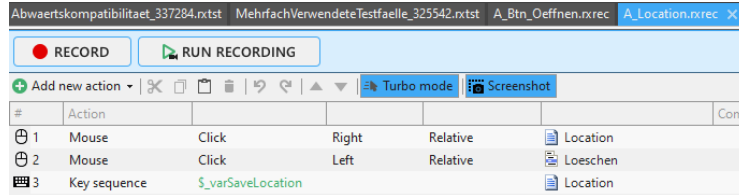


Figure 6.4: Actions and Events

An advanced way of implementing changing is through writing CSharp codes. To fulfill this, we first had to create and bind a variable and assign to a specific test suite where it is going to be used. This technique also involved programming the GUI features. To get any feature from the software, Ranorex Spy was used so that the feature could be accessed through CSharp code. The spy feature of Ranorex is shown in the figure 6.5.

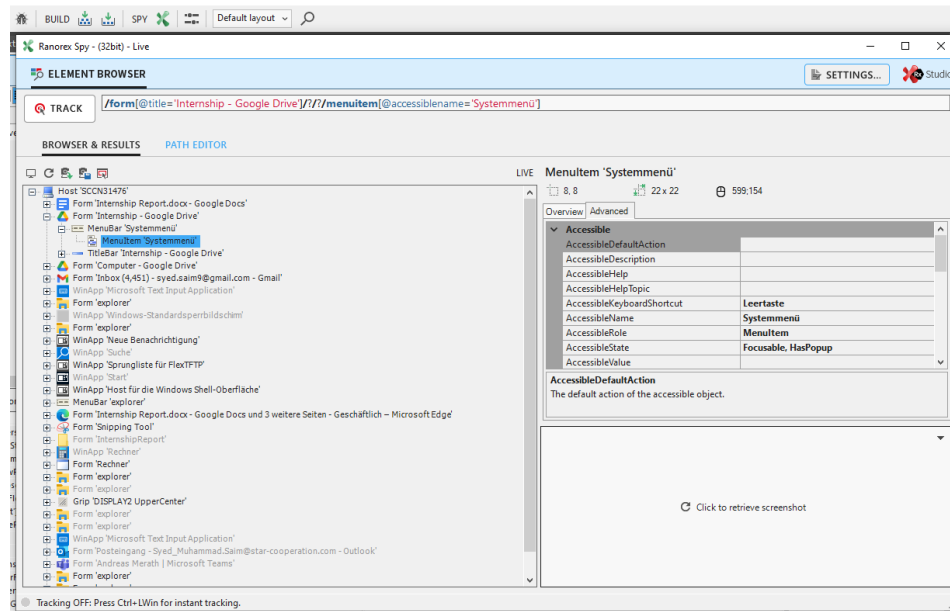


Figure 6.5: Ranorex Spy

A button in the ranorex toolbar was clicked that opened ranorex Spy. As the track button was clicked, we enabled the Ranorex Spy to detect any feature of the Application Under Test (AUT). By hovering the mouse over any icon or feature of AUT, it also showed a red box for that specific area where the mouse is detecting and required icon was clicked. After detecting, the address was assigned for that icon in the bar next to the track button. This address could then be used in CSharp code, hence making it accessible.

Once the feature is recognizable, it could be accessed through writing code in CSharp. As the software Ranorex was testing Flex Config RBS software of the company, there were many features that were failing some of which could not be upgraded through only recording or disabling any action or event. For example a dialogue box appeared while testing which was not there previously. Mouse had to click the 'ok' button if this dialogue appeared so the code had to be written. To complete the task effectively, I wrote the CSharp code in which I used try catch statements as shown below:

```

1 try
2 {
3     Form selectFamilyConverterDialogue = "/form[@controlname='
4     SelectFamilyConverterDialogue']";
5     ecu_dialogue_visible = selectFamilyConverterDialogue.Element.
6     Visible;
7 }
8 catch
9 {
10     ecu_dialogue_visible = false;
11 }
12 if (ecu_dialogue_visible)
13     break;

```

Listing 6.1: CSharp

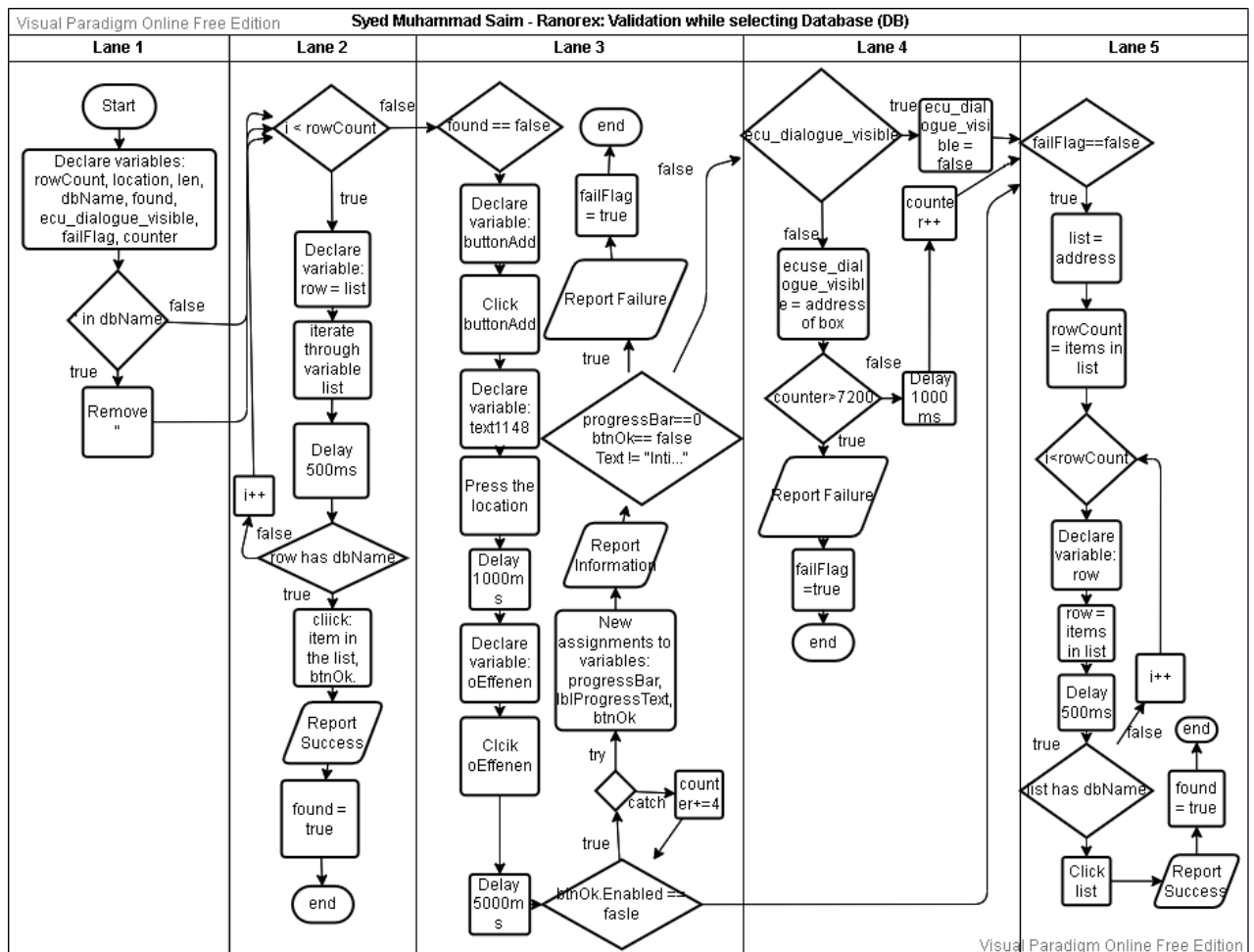


Figure 6.6: FlowChartDiagram for the written code

Chapter 7

INTERNSHIP EVALUATION

7.1 Academic Relevance

Many of the concepts related to this industry were related to my study course. I got to know the practical picture of what I had learned at university based on theory. For instance, I learned and wrote a research paper on Controller Area Network (CAN) which helped to understand communication protocols. The concepts of IP address and TCP/UDP, which I studied in the third semester, also helped to gain a good understanding of the advanced concepts. In addition, concepts of Hardware Engineering also helped in getting what is going on while I was testing the hardware of the company in the development department. Lastly, programming skills which I learned throughout my study course assisted me in understanding the huge professional codes.

7.2 Skills Learned

During my internship tenure, I learned a number of advanced concepts that polished my skills further. Firstly, I got the opportunity to understand a professionally developed code in CSharp programming language. I tried to debug the code based on the given task and upgraded the code by adding more functionalities. This gave me an advanced experience by enhancing my coding skills. Moreover, I got to work several software such as Ranorex, which gave me the knowledge of automating the testing of any software. Furthermore, I learned theory of various automotive bus systems and communication protocols that are being used in automotive vehicles. In addition, I got to attend several scrum meetings with other colleagues on weekly bases. Moreover, I experienced team work while using tools such as Git and SVN tortoise. This particular experience gave me an idea that how branches and versioning are actually functional. Apart from it, I got an experience in dealing with different colleagues who came from a different study backgrounds. We shared various ideas of innovation. Experiencing such an environment has enhanced my soft skills which are definitely relevant for future employment perspective.

7.3 Pros And Cons of the Internship

Doing internship at Star Cooperation GmbH was a worthy experience. It gave me a professional experience that how a company is divided into various tasks and all of the task are working independently while collaborating with each other. Initially, it was quite tricky to understand the overall workflow of the company but as time passes and as I was shifted to various departments, I came to know various areas of application within the company. The whole decorum of the company is peaceful which helps to focus more on the tasks. Colleagues are quite friendly and always ready to help. Working here allows me to get to know a big picture of what is going in the world related to different kind of technicalities and how the different companies are working hand-in-hand. Moreover, it was a great experience to know how different softwares are used in a unique way during the development of a product. So it not only the effort of people and companies but the effort of various softwares that were employed to do a specific job. I had learned a lot at university but applying those concepts in a professional world requires tremendous effort.

Apart form it, it is hard to think any drawback of the internship phase. A drawback which can be highlighted is that there were times when I did not have any tasks to do but I consider this part as an opportunity and learned more about the company through reading documentations.

*Appendix A***CONSENT FORM**

Declaration of Authenticity

With my signature I hereby declare that I have developed my part of this project by my own and did not use any unnamed and/or forbidden sources or aid.

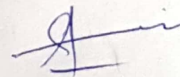
In particular, no code was taken from other sources than the lectures.

I know that plagiarism is a criminal offense.

Name (readable): Syed Muhammad Saim

Date 15.08.2022

Signature (handwritten)



BIBLIOGRAPHY

- [1] Boris Beizer. *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc., 1995.
- [2] STAR electronics GmbH Co. KG. Flextinys documentation. *IEEE transactions on industrial electronics*, 54(4):1824–1842, 2016.
- [3] Ilya K Ganusov, Mahesh A Iyer, Ning Cheng, and Alon Meisler. Agilex™ generation of intel® fpgas. In *2020 IEEE Hot Chips 32 Symposium (HCS)*, pages 1–26. IEEE Computer Society, 2020.
- [4] DHEERAJ KAKARAPARTHY. Overview and analysis of automated testing tools: Ranorex, test complete, selenium, 2017.
- [5] Gavein Mogan Kohsuke Kawaguchi. Jenkins workflow. *jenkins.io*, 110(29):11982–11987, 2013.
- [6] STAR Electronics GmbH Co. KG. Controller area network. *star-cooperation.com*, 25(11):2886–2895, 2016.
- [7] STAR Electronics GmbH Co. KG. Controller area network - flexible data-rate (can_fd). *star – cooperation.com*, 199(2) : 391 – –401, 2016.
- [8] STAR Electronics GmbH Co. KG. Flexconfig rbs software documentation. 66(3):254–272, 2016.
- [9] STAR Electronics GmbH Co. KG. Flexdevice-l documentation. 100(1-3):163–178, 2016.
- [10] STAR Electronics GmbH Co. KG. Flexdevice l² documentation. 39(10):5280, 2016.
- [11] STAR Electronics GmbH Co. KG. Flexdevice-s documentation. 390(10113):2627–2642, 2016.
- [12] STAR Electronics GmbH Co. KG. Flexray. *star-cooperation.de*, 3(2):125–138, 2016.

- [13] STAR EElectronics Gmbh Co. KG. Introduction. *star-cooperation.com*, 2:147–171, 2016.
- [14] STAR Electronics GmbH Co. KG. Scalable service-oriented middleware over ip (some/ip) documentation. 69(11):13450–13466, 2016.
- [15] Akira K Onoma, Wei-Tek Tsai, Mustafa Poonawala, and Hiroshi Suganuma. Regression testing in an industrial environment. *Communications of the ACM*, 41(5):81–86, 1998.
- [16] SOURCEFORGE.NET. Tortoise svn workflow. *tortoisesvn.net*, 101(2):020403, 2016.