Development of an Automated Tool for Valve Driver Configuration

Divya Janki Patel, Nithin S, Sivaprakash N

***Abstract*: The development of a software tool for automating configuration of valves used in automobiles is discussed here. Configuring the valve driver is a monotonous task requiring humongous amount of time and concentration. The tool extracts data from Microsoft Excel file and generates the source code for controlling valves. The tool features a user friendly GUI which allows the user to input and import the configuration files.**

***Index Terms*: Automation, automotive valves, safety critical.**

I. INTRODUCTION

Automotive valves are used for a variety of functions extending from windscreen wiper, automatic door locking, side screen control, cooling, suspension control [1], steering control, fuel circulation, primary braking, secondary braking and emergency braking. The performance and responsiveness of different systems of the vehicle directly depend on the automotive valves [2], [3], [4]. Manufacturers have developed highly sophisticated valves to address the increasing complexity of engines. These valves can considerably increase the safety and efficiency of the vehicle and also match the performance and luxury demand of the consumer. The device driver for valves are modeled to be compatible with a layered architecture similar to AUTOSAR architecture [5], [6] called CESAR [7]. This increases the complexity of the driver as all the signals in a layer must be mapped to other layers. On an average any specified automotive project could easily have 15 valves. Approximately 110 parameters are to be configured for these valves. Generating code for such complexity may take days as 2000 parameters are to be configured. The valve driver is manually configured and hence prone to human errors, furthermore it is a tedious and time consuming process [8]. The complexity of configuration demands the developer to critically analyze the software for bugs. Automation of industrial systems has demonstrated increase in efficiency and productivity [9]. Creating a tool to automate code generation [10] for embedded domains helps to save time when compared to manual coding. The automation proposed here involves the development of a software tool to automatically promote code generation that was previously written manually.

Revised Manuscript Received on July 06, 2019.

Divya Janki Patel, Department of Electrical and Electronics Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidhyapeetham, India.

Nithin S, Department of Electrical and Electronics Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidhyapeetham, India.

Sivaprakash N, Basic Software, WABCO India Ltd, Chennai, India.

II. VALVE DRIVER CONFIGURATION

The functionality of a valve driver is depicted in Fig. 1. One function is to actuate the valve and the other is error detection and related safety reaction. Detection of errors in a valve is of utmost importance as valves are used for safety critical applications. Frequent checks need to be performed on valves to ensure safe engine working [11] which in turn leads to the safety of passengers.

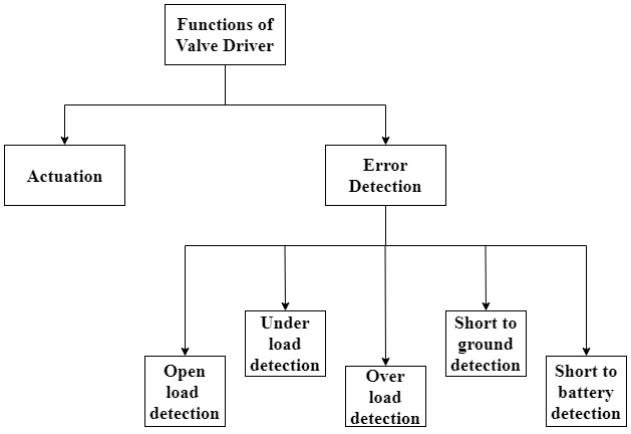

Fig. 1. Functions of Valve Driver

The organization of parameters in signal configuration is based on the signal name, initial state of the signal, power supply information, duty cycle, verification of control switch and many such parameters. Similarly some of the parameters in error configuration are feedback signal name, Error id, mapping error detection. Typically there are such 64 parameters for error configuration and 30 parameters for signal configuration. Apart from these there are other configurations that give us detailed information on disparate parameters such as number of blocks, names of the block, instance of blocks and also configurations related to power supply. All of these parameters are then to be incorporated in to the device driver for the accurate functioning of the valve.

III. DEVELOPMENT OF CONFIGURATION TOOL

To automate the code generation for valve component, a software tool named *Valve Configurator* is developed. The input to the valve configurator is an excel sheet that has all the requirements.

It consists of all the data relating to signal configuration, error configuration and supply configuration of valve that is functionality specific as shown in Fig. 2.

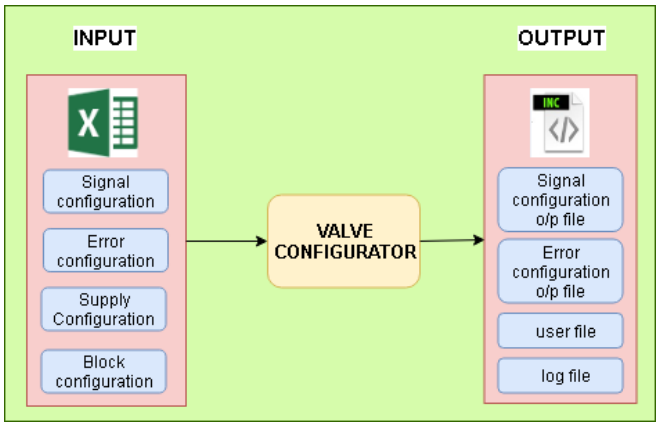

Fig. 2. Block Diagram of Tool

The excel input is a requirement document consisting of ten sheets that has several configurations namely signal, error, supply and other such information that is related to the working of solenoid. The valve configurator then reads this input excel sheet and takes the necessary data and stores it in a variable. This data is then manipulated in such a way that it can be used in a layered architecture as the source code generation is carried out for a complex driver and generates *.inc* format output files explicitly for signal and error configuration, patches a *user.inc* file and also creates a *.txt* file which stores all the collected data for any future clarifications. The valve configurator is programmed using object oriented programming techniques as it provides modular structure, easier to maintain and is reusable. It is also helpful during the development of script, as the user need to alter the class attributes and include/remove methods thus making development process speedy and effortless. The complexity of programming the automation script is also decreased to a large extent. Python is used for the development and the configuration files to be auto generated could be user defined. It is easy to use and is not affected by windows update unlike Visual Basics for Applications and MS Access. Script development is done in python using Eclipse. Some libraries used for the development are openpyxl, Thread, web browser, tkinter, datetime, time, shutil, os and sys [12]. The valve configurator also has a graphical user interface for selecting the input requirement sheet and also to give other mandatory files necessary for code generation. It also has a status bar that indicates the working of the tool and also displays messages if any incorrect action is taken by user. It also validates the format of excel sheet and works if the project is compatible with the given version in tool. If not a message is displayed on status bar.

IV. IMPLEMENTATION

The valve configuration files are usually written and compiled in Eclipse, the tool developed can be easily integrated to Eclipse as an add-on. It launches a GUI with the title as project name and two tabs as depicted in Fig.3. The first tab in GUI consists of six labels, six buttons and a status bar. The second tab in GUI consists of three buttons that are used to open user manual, email to clarify queries and a Google sheet to record problems faced using the tool on a click respectively. The first tab of GUI consists of label specified for each button describing its usage. The valve configuration process is depicted in Fig.4.

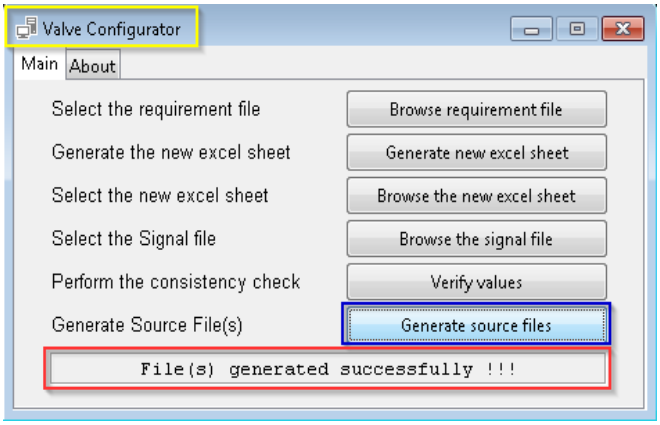

Fig. 3. User interface of valve configurator

The tool has six button, their functionalities are briefly explained below.

1) Browse requirement file : The button is used to browse the requirement sheet from the system.
2) Generate the new excel sheet: This button creates a new excel sheet where in the developer fills in the parameter values in the sheet that may not be present in the original requirement sheet. To avoid making changes to the actual requirement sheet, a new sheet is created and values are given to the parameters so that these new values could be replaced by old ones without affecting the original requirement sheet.
3) Browse the new excel sheet: This button is used to browse this new sheet that is just filled by the developer.
4) Browse the signal file: This button is used to browse a file from the project that has all the valve signals in a given order. To avoid any compilation errors we browse the file by clicking on the button called browse the signals file and generate the code for the valve signals according to the file.
5) Verify values: This button is to verify if any parameters values that are to be given by developer are present. Once the check is done and no values are found the status bar in GUI changes from system name to “verifying done”.
6) Generate source files: This button is used to generate the source files. Once the files are generated the status bar changes to “files generated!!! “

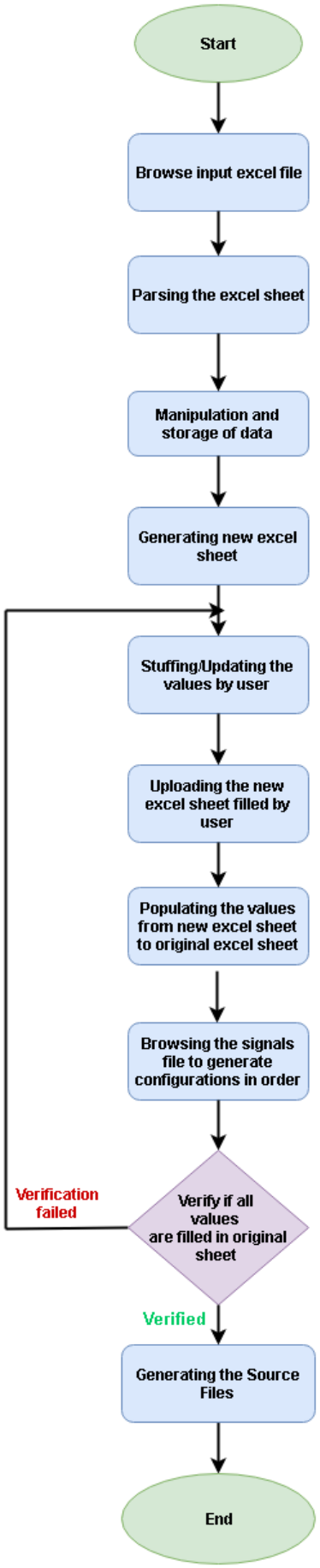

Fig. 4. Process of valve configuration

V. RESULTS

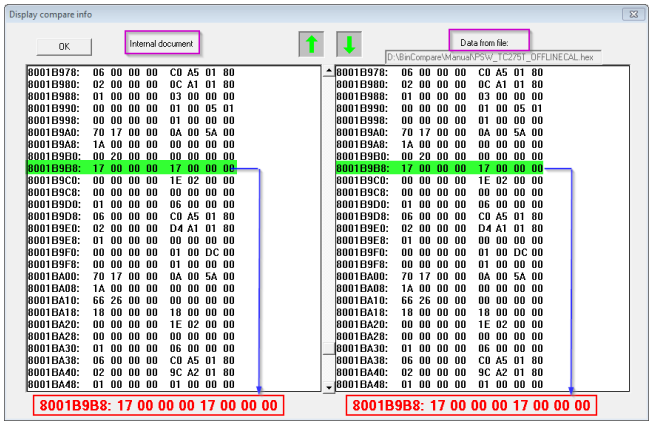

Fig. 5. Output from the tool vs. manual configuration

Fig. 3 shows the GUI and the status bar that indicates that files are generated successfully, when the generate button is pressed. The hex comparison of the source code that was manually written to that of the source code which was self-generated using the valve configurator is shown in Fig. 5. Hex view tool [13] was used to perform the hex compare for both the codes.

VI. CONCLUSION AND FUTURE SCOPE

The valve configurator tool developed decreases the time required to configure the valves from days to just minutes. The complete process of understanding the tool and using it for the first time for a project would require 30 minutes. However subsequently for the same project it would take less than a minute as the new excel sheet is already filled by the user. The future scope includes the automation of the requirement generation and testing of the signals generated by code in oscilloscope.

REFERENCES

1. W. Ribbens, Understanding Automotive Electronics. Butterworth-Heinemann, 2017, Fifth Edition.
2. Goszczak, Jarosław & Radzymiński, Bartosz & Werner, Andrzej & Pawelski, Zbigniew. (2017). PWM-controlled hydraulic solenoid valves for motor vehicles. The Archives of Automotive Engineering. 75. 23-37. 10.14669/AM.VOL75.ART2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. Xu, Xiangyang & Han, Xiao & Liu, Y.F. & Liu, Yanjing & Liu, Yang. (2017). Modeling and Dynamic Analysis on the Direct Operating Solenoid Valve for Improving the Performance of the Shifting Control System. Applied Sciences. 7. 1266. 10.3390/app7121266.
4. Passarini, Luis & P. R, Nakajima. (2003). Development of a High-Speed Solenoid Valve: An Investigation of the Importance of the Armature Mass on the Dynamic Response. Journal of the Brazilian Society of Mechanical Sciences and Engineering. 25. 10.1590/S1678-58782003000400003.
5. Daehyun Kum , Gwang-Min Park , Seonghun Lee and Wooyoung Jung "AUTOSAR Migration from Existing Automotive Software" International Conference on Control, Automation and Systems 2008 Oct. 14-17, 2008

6. Daehyun Kum, Jangkyung Son, Joonwoo Son and Myungjin Kim, "Automotive Embedded System software Development and Validation with AUTOSAR and Model-Based Approach," Journal of Control, Automation, and Systems Engineering, Vol. 13, No. 12, pp. 1179-1185, 2007.
7. Kalix, E. and Schuette, O. (2007). CESAR An architecture for software development networks - Technische Informationsbibliothek (TIB). [online] Tib.eu. Available at: https://www.tib.eu/en/search/id/tema%3ATEMA20070800938/CESAR-An-architecture-for-software-development/.
8. "Toyota "Unintended Acceleration" Has Killed 89", May. 25, 2010.[online]. Available: https://www.cbsnews.com/news/toyota-unintended-acceleration-has-killed-89/. [Accessed : May. 10, 2019]
9. S. D. Shenoy, Sharma, V., and S .Santhanalakshmi, "Automation of Timetable Generation using Genetic Algorithm", Int.J.Computer Technology & Applications, vol. 5, pp. 1491-1494, 2014.Ib
10. F. Jose and Pillai, A. S., "Code configuration tool for real time systems", in 6th International Conference on Computation of Power, Energy, Information and Communication, ICCPEIC 2017, 2018, vol. 2018-January, pp. 342-346.
11. Y. C. Nair, Kumar, S., and Dr. Soman K. P., "Real-time automotive engine fault detection and analysis using bigdata platforms", Advances in Intelligent Systems and Computing, vol. 515, pp. 507-514, 2017.
12. "The Python Standard Library", June. 03, 2019.[online]. Available: https://docs.python.org/3/library/. [Accessed : June. 07, 2019]
13. "Hexview", [online]. Available: https://cos.colorado.edu/CEDAR/CEDAR_HELP/hexview.html. [Accessed : June. 18, 2019]

AUTHORS PROFILE

Divya Janki Patel, Department of Electrical and Electronics Engineering, M.Tech, Embedded systems, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India.

Nithin S, Assistant professor, Department of Electrical and Electronics Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India.

Sivaprakash N, Senior Team Leader, Basic Software, WABCO India Ltd, Chennai, India.