Expertise
and insight
for the future

Jorma Pohjola

Design Rules Manual for Volvo Multiplex Toolbox

Metropolia University of Applied Sciences

Bachelor of Engineering

Automotive Electronics

Bachelor's Thesis

02 December 2019

Author	Jorma Pohjola
Title	Design Rules Manual for Volvo Multiplex Toolbox
Number of Pages	17 pages
Date	2 December 2019
Degree	Bachelor of Engineering
Degree Programme	Automotive Engineering
Professional Major	Automotive Electronics Engineering
Instructors	Eero Kalevo, Business Unit Manager Vesa Linja-aho, Senior Lecturer

In modern world, networking and data sharing enable corporations to spread out design work. The creative work in designing new products, such as a bus, can be spread between many teams. In some cases, these teams, and in fact members of a team, might be separated by continents. It is possible for one to have close working relations with people that they have never met in person.

In this kind of environment, standardization of practices and uniformity of output is more important than ever. To this end, multinational entities create manuals and rules to make it easier for workers around the globe to keep their creations such that others can easily understand them.

Volvo Multiplex Toolbox, abbreviated VMT, is a design tool suite used by Volvo Bus Corporation in making the software packages that control behavior of the bus body systems. This software package combines needed tools to create, modify and share different software components used for making software packages that are then installed in completed vehicle. The software is created using Functional Block Diagram-language, a graphical logic programming language.

In this thesis, a rules manual was created, and it works both as introduction for new employee on how different parts of VMT work and creates ground rules on how to design new body software. Much importance is placed on creating rules for the graphical look of the final code, and uniform naming policies on different signals, components and parts.

Keywords	Function block diagram, Automotive electronics, User manual

Contents

List of Abbreviations

BEA	Bus Electronic Architecture. Describes the generation of electronic architecture in a Volvo bus.
CAN	Controller Area Network. Message-based communications protocol used extensively in automotive industry.
DTIC	Distributed Target Independent Code.
FBD	Function Block Diagram. Graphical PLC-programming language described in IEC-61131-3-standard.
PLC	Programmable Logic Controller. Ruggedized and simple computer used in industrial applications that don't need much computing power.
VMT	Volvo Multiplex Toolbox. Software suite used for creating body software packages for Volvo buses.
ORM	Object-relational mapping. The set of rules for mapping objects in a programming language to records in a relational database, and vice versa.
DBMS	Database management system. Software for maintaining, querying and updating data and metadata in a database.

1 Introduction

Thanks to the spread of internet, new ways of networking and sharing of data across the world is possible in ways that were unthinkable just a couple of decades ago. Information, ideas and digital products can move between continents in seconds, and people can connect and communicate without needing to think about distances.

In a modern, multinational workplace, networking allows workloads to be spread over a wide area. It is not necessary for a team to go to one place to design a new product. Especially for large corporations that function globally it is much easier and cheaper to spread the work to be done across multiple locations. Moving personnel and equipment is expensive, moving information costs next to nothing.

This kind of spreading the workload raises new kinds of problems. Members of a design team come from different cultures. They have different backgrounds and they speak different languages. They think differently and solve problems in different ways. How is it then ensured that people who might have never even met in person can understand each other?

To solve this problem, businesses create rules and procedures that are implemented in same way everywhere. No matter whether you work in Finland or India, you conduct your work in the same way: Use the same document forms, use the same abbreviations and the same tools with the same names.

To spread these rules around, you need manuals that contain the rules in a form that is easy to digest. Manuals should be made so that new employees find in them easy to learn what to do and how, and so that professionals that have been working for years can easily refresh their memories where needed.

In this thesis, creation of one such manual is examined, made for Volvo Bus Corporation (VBC). The aim of the manual is to uniform the workflow, naming conventions and coding practices used with Volvo Multiplex Toolbox (VMT), a software suite created by Microteam Oy and used by VBC to create software packages for bus body systems.

1.1 Microteam Oy

Microteam Oy was a Finnish engineering company, now a part of Insinööritoimisto Comatec Oy. Founded in 1981 and based in the city of Tampere, Finland, Microteam offered product and software development and engineering services in areas of industrial electronics, machine automation, robotics, and vehicle control systems. (1)

Microteam Oy was merged with Insinööritoimisto Comatec Oy in July 2019.

1.2 Volvo Bus Corporation

Volvo Bus Corporation is part of Volvo Group, a global manufacturer of trucks, buses, construction equipment and marine and industrial engines. Volvo group has close to 100 000 employees worldwide and sells its products in more than 190 different markets.

Volvo Buses develops, manufactures and sells complete buses and bus chassis for city, intercity and tourist operations. Offerings include electric city buses, hybrid buses and diesel buses in different power ratings. Volvo Bus Corporation includes four different brands: Volvo, Nova Bus, Prevost and UD. (2)

1.3 Volvo Multiplex Toolbox

VMT is a suite of programs, created by Microteam and first released in 2005. It was first made as a proof-of-concept-software for testing new hardware ecosystem but ended up being used as-is for over a decade. The current version, VMT2, was released in 2015 and is actively being further improved and updated.

VMT contains all the necessary tools for developing and creating all the application software and hardware-related software components used in Volvo buses for body system control. This includes necessary definitions for input/output (I/O) of electronic control modules (ECUs), hardware-software-interface definitions and function block diagram-language compiler for creating application software for different body functions.

All the software components are stored in cloud storage, so that all users of VMT have the same components available to them. Access to the cloud storage for saving and distributing software components between users is implemented in VMT. The cloud storage is part of Volvo IT, but it is managed and administrated by an administrator team based in Finland.

In addition to Volvo Bus Corporation, Prevost Car and Nova Bus, both based in North America, use VMT for developing body software.

2 Design Rules Manual

2.1 Basis

Before this project, there had been no unified manual about using VMT. Users and developers of VMT had made some short and incomplete manuals on different parts of the software during the past decade. Many of these documents were so much out of date as of being nearly unusable. Others were not usable alone, describing just one part of software or one process.

There were also many technical aspects of VMT, especially on hardware-side, that were glossed over in older manuals. These parts required a lot of research in Volvo documentation libraries to find more about hardware specifications, as more information was needed to answer questions about why software interfaces with hardware as it does.

Training new users has been on hands of colleagues to be done as they see fit, with manuals and rules made in each different office doing BEA-body software design around the globe. Different ways of working with the software was starting to show as software components with different levels of documentation and style of programming. This causes rising costs, as interpreting premade parts of software was taking longer.

In the end, VBC decided to rectify the situation by ordering creation of a complete manual describing the basic usage of VMT and setting down a set of rules to govern the software development for Volvo buses.

2.2 Requirements for the Manual

Basic requirements given to the manual were:

- To be usable as a training and support manual for new employees working in software component design, and as a reference manual for senior designers.

- To be extensive enough to cover most use cases of VMT, including installation, component creation and modification, database management and post-process compilation of complete body software packages.

- To contain all the necessary rules and conventions to unify software component creation across VBC offices, including naming conventions and styling of graphical function block diagram-code.

- To contain information about solving common program compilation errors that a designer might encounter.

- To contain real use case examples of component creation and modification, and

- To contain information about hardware components, how they interface with software and how to find necessary information about input and output options, in as much detail as is necessary for software component development.

2.3 Scope of the Manual

Despite the name, the finished product is much more than rules library for designing software. It is also a user manual for the whole VMT-software and most support functions tied to it. The aim is for the manual to be written in such a way that a new employee can use it as a guide to easily start working with the software.

Due to being written by someone who had no prior experience using the software, the scope of the manual was not apparent at the start of the work. While learning more about the usage of VMT, more and more chapters were being added in the manual.

When this bloat of information was starting to be noticed, it was decided that some parts that were included in older manuals made for different parts of the software were to be left out. Things that had nothing, or very little, to do with people working with design and more to do with the field work at the Volvo workshops or manufacturing plants were glossed over briefly or completely cut out. This had to be done to keep the workload manageable.

Despite cutting out parts of old manuals, the complete manual contains more information than requested in the initial requirements. It seemed that this was necessary to give designers enough in-depth information about underlying principles of VMT to enable effective management of workflow.

One of the main goals given for the design rules manual was to create unified style for developing the Function block diagram-code used in Volvo BEA-systems. There have previously been rules about naming and conventions about visual style and commenting of programs. However, as the software component development has been branched out between multiple different offices, it was deemed necessary to compile the official manual codifying all the rules and the established conventions. The manual was to have examples on how to implement the rules mentioned earlier.

3 Function Block Diagrams and Bus Body Control

3.1 What are Function Block Diagrams?

Function Block Diagrams, abbreviated FBD, are one form of graphical languages used for programming Programmable Logic Controllers (PLCs). These kinds of ruggedized, simple digital computers are widely used in many kinds of control tasks in industrial environment. (3, p. 3-4)

FBD is one of two graphical logic programming languages defined in IEC-61131-3-standard. In addition to graphical languages, this standard describes two text-based languages to be used with PLC programming. (4, p. 9) The standard works as a guideline, not ruleset, for PLC-programming. (5, p. 12)

While IEC-standard defines syntax of PLC-programming languages using text-based instruction sets, usually FBD-programs are created in graphical environment for simpler user experience. Exact look of this graphical user interface is not standardized. (4, p. 211-212) An example of simple graphical FBD-program taken from VMT programming environment can be seen in Figure 1.

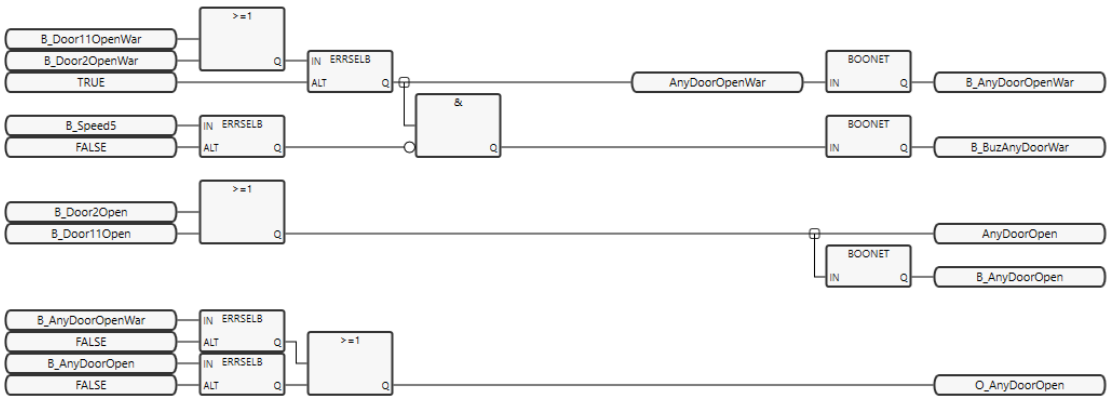

Figure 1 Example of FBD-program, taken from VMT enviroment.

Basic premise of FBD-programming is using inputs, coming from the left side of the program, and feeding them through logical functions, drawn as rectangular blocks, achieving outputs on the right side of the program. Programs execute, or flow, from left to right and from up to down. This results in that the arrangements of the blocks are important

when creating the code. Creating situations where signal paths go wrong way, can cause unexpected behaviour or compilation issues.

Logical functions that can be used with FBD-programs vary from simple Boolean algebraic functions such as AND, OR and NOT, to simple set-reset-latches and timers, to more complex user-created functions such as networking control and error correcting algorithms. How these more advanced user-created functions get implemented vary between PLC-vendors and different programming environments. Some systems also support processing of analogue values, but this requires compatible hardware.

3.2 FBD in Bus Body Control Systems

In the automotive industry, cost efficient manufacturing and design processes are one of the leading elements of today. To this end, body systems controlled with computers programmed using PLC-like systems and FBD-logic offer great benefits.

In general market, bus manufacturers usually choose between two commonly available competing control systems with ability to provide control and communication for a whole body system: Thoreb ELSY offered by Thoreb Ab, and KIBES created by Continental Automotive GmbH. There are also manufacturer-specific systems, such as Volvo Body Multiplex, that are not available on open market.

In bus market, manufacturers see two different kinds of customers: large companies offering mass transport with large fleets of identical vehicles, and usually smaller businesses offering coach services. These two kinds of customers require quite different kinds of vehicles, with different level of customisation and standardisation. Suitably flexible control system that is easily modified to suit customers' needs is a great marketing tool for any manufacturer.

Using PLC-like systems with added Controller Area Network (CAN)-capabilities for flexibility offers possibilities for easy customer-specific tailoring of functions. Creation of standardized software packages allows for re-using created code for faster development process.

Customer adaptations are also easier and faster to create when standard programs can be used as a base and are simply modified. As an added benefit, with flexible enough interface, modifications to body functions can even be made many years after the vehicle has left the manufacturing plant: this increases the resale value of the vehicle, making it more enticing purchase for the original buyer.

4 Process of Creating Design Rules Manual

4.1 Introduction, Installing and Setting up the Program

The manual creation process was started by finding out basic information about the program: why and when VMT was made, what the principles were when designing it. This was done to give the reader some perspective on the history of VMT and Volvo Bus Body design, information about different services and support functions, and basic information about different user levels.

After the short introduction, steps on installing and setup of the program were given. This includes where to find the installation package, where to request access to necessary databases and how-to setup the necessary folder structure by the user. These instructions were accompanied by multiple screenshots to make the process clear and simple. Process of and importance of daily updating the databases used on handling different software components was described in-depth.

4.2 Component Description

The manual describes all the different software components it is used to create. These short descriptions give basic information about function and importance of each different type of component.

4.3 Software Package Generation

Manual includes a chapter about back-end processes that are invisible to a basic user of VMT but are of great importance for the process of automatic generation of basic body software packages from pre-made, standardized software components. This information is useful for end users who work with aftermarket support processes and need to be able to do problem solving with limited resources.

4.4 Software Component Design

Part of the user manual concerning the day-to-day usage of VMT. This includes creation and upkeep of software components and how to distribute the created components to other VMT-users.

The manual includes description and use case of every different component used in a complete Volvo Bus body software package. There are also descriptions of related hardware-software-interfaces and instructions how to use them to help with troubleshooting and find more information when encountering problems with software development.

4.5 FBD Design Rules

The original request for the thesis work was to create rule package for styling of graphical function block diagram programs created with VMT. Before starting, the scope of work was extended to the whole manual. This part of the manual is closely associated with software component creation.

More detailed description about the creation process and examples of rules can be found in chapter 5.

4.6 Troubleshooting and Debugging Information

Information for troubleshooting and debugging created programs is included for most common problems encountered. Examples with screenshots showing some of the possible errors and ways to fix them are given.

As a small amount of information about troubleshooting problems with full software packages downloaded to a vehicle are given. This is, however, kept to a minimum as this kind of diagnostic work is detailed in other manuals.

4.7 Information about Hardware

Some information about the physical components of the body system is given, as knowing the limits of physical input and output capabilities of available hardware is needed when designing programs. Knowing, in example, limits of output current of the module when designing lighting or blower motor controls is extremely important for designers.

4.8 Practical Examples of using VMT

In the last part of the manual, practical use-case examples of VMT were created. For these, specifications for imaginary customer adaptation order for body system were created and according to these, software components were created and tested.

5 Guidelines for FBD-programming

5.1 Function Block Diagrams in VMT

At the heart of the BEA-body software are software functions executing programs created using a graphical FBD-programming environment. More information about FBD can be found in chapter 3.

During initial development of VMT and its accompanying ecosystem of ECUs, it was decided that FBD offered best possible combination of necessary features and ease of use and it was implemented as part of VMT. Inspiration was drawn from Thoreb ELSY-system. Originally, ISaGRAF-editor by Rockwell Automation was used for FBD-programming but eventually own, in-house developed programming and compilation environment was created and ISaGRAF was phased out.

FBD-code created with VMT is compiled in Distributed Target Independent Code-format by compiler built-in with VMT. The control units installed to the vehicle are listed for the compiler in the code and using this list code is distributed to the control units during compilation process. This compiled code package is then downloaded into bus system and used to control body functions.

During the pre-work investigation on the matter, it was found that there is no publicly available collection of practices or styling guides for FBD-programming. It was, therefore, up for the writer to codify the rules for FBD-programming for VBC from scratch.

5.2 Styling Needs and Best Practices

Finding out the best practices in making FBD-programs involved interviewing people that have been making programs for years. As FBD is a graphical language, some amount of artistic expression is evident in each interviewee's personal style.

Easy readability of the code was the number one requirement of every coder interviewed. Positioning of function blocks and signals is important and was emphasized heavily. By

keeping the signal lines in the code straight, having fixed places for inputs and outputs and lining function blocks straight makes code easier to read, and pleasant to look at.

Debugging the code is easier when signal naming is simple and follows established conventions and guidelines.

Some technical needs for styling arise from the development process. For example, sometimes code needs to be printed on paper in order to facilitate easier debugging or going through parts of the code in meetings. To minimize clipping of the code, it needs to be kept narrow enough to be able to be printed on A3-sized sheets.

Commenting code, especially more complex parts, was mentioned as an important part of the programming process. This makes comprehending the code faster and makes reusing existing functions easier. Using ability to change the colour of parts of code implemented in VMT is also important, as marking modifications done to code using colours speeds up possible debugging, especially when done by other developers.

5.3 Finding and Creating Examples

For some requirements, examples were easy to find. Especially the older code, created well before any kind of unified system for signal naming or placement of logical elements in the code were made, offered ample amount of examples on how not to create programs. Changes in VMT has also affected some of the older software components, making the visual elements of programs not looking right. An example of this kind of a problem can be seen in Figure 2.

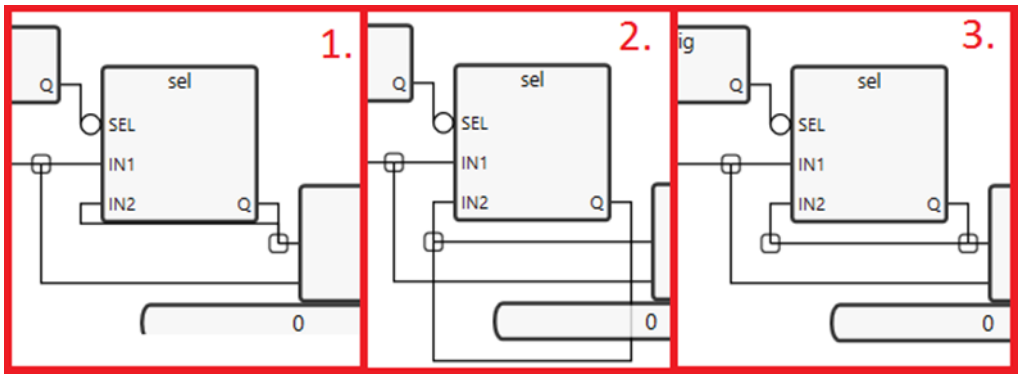

Figure 2 Styling problem example used in finished manual

In some cases, suitable programs with correct problems were not easily found. When this happened, an example was created.

5.4 Naming Conventions

Naming rules for input and output signals was an important issue to underline in the manual. Naming of signals directly constitutes to the ease, or unease, of reading the code made by someone else. Descriptive naming makes the function of inputs and outputs of code easily recognized, while signals named in just alphanumerical strings makes it hard for others to decipher the intention of the original creator.

Sometimes one program can have well above hundred different input and output signals, from both physical hardwired components and from other control units in form of data signals from CAN-bus, in addition to private variables. If naming of the signals is not done according to uniform style, interpreting code will become extremely hard.

To make it easier for designers to name new signals, a simple table with preferred prefixes and shorthand for common components, such as switches, buttons and doors, was included in the manual. It was also emphasized that using existing signals is preferred to creation of new ones whenever possible.

6 Conclusions

In conclusion, the main objective of the thesis work was achieved: the customer received a new manual containing necessary information about usage of Volvo Multiplex Toolbox in body software creation, and a new set of rules was created for styling of graphical function block diagram programming.

References

1 Microteam, company introduction. 2007. Internal document, Microteam Oy.

2 Making a difference – Moving people. Volvo Buses corporate presentation. 2019. Internal document, Volvo Buses Ab

3 Bolton, W. 2009. Programmable Logic Controllers, fourth edition. E-book. Elsevier Science & Technology.

4 SFS-EN 61131-3. Programmable controllers – Part 3: Programming languages. 2013. Helsinki. Finnish Standards association.

5 John, Karl-Heinz; Tiegelkamp, Michael. 2010. IEC 61131-3: Programming Industrial Automation Systems. E-book. Springer-Verlag Berlin Heidelberg