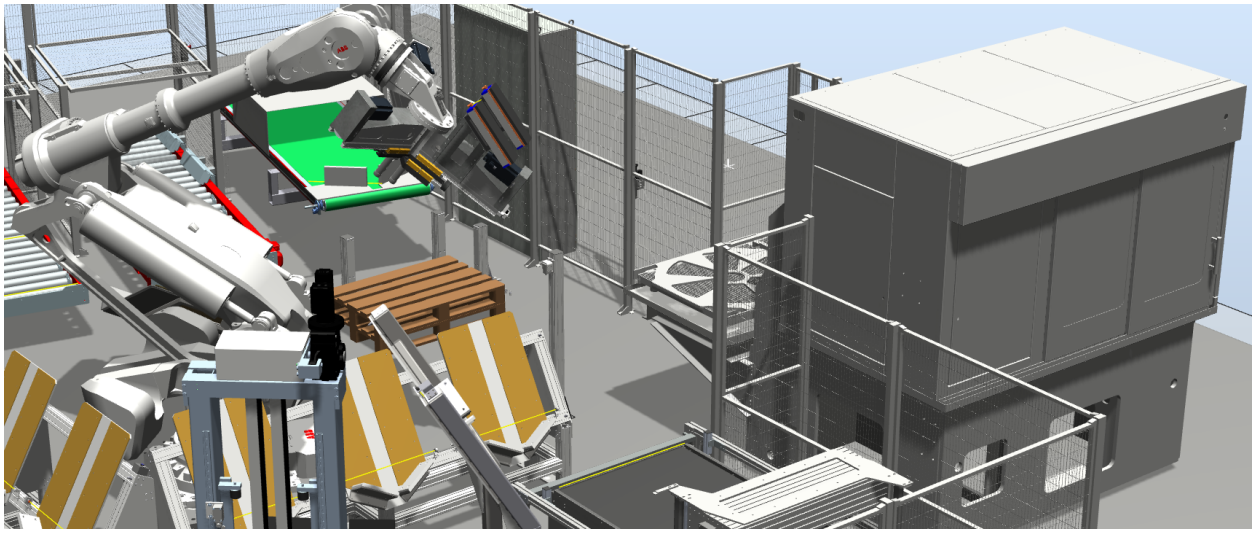
CHALMERS
UNIVERSITY OF TECHNOLOGY



Efficient modelling techniques for virtual commissioning

Signal communication in robot cell using SIMIT templates

Bachelor's thesis in Mechatronics

GUSTAV GUSTAFSSON
MARTIN GUSTAVSSON

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Bachelor's thesis 2019:06

Efficient modelling techniques for virtual commissioning

Signal communication in robot cell using SIMIT templates

GUSTAV GUSTAFSSON
MARTIN GUSTAVSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Efficient modelling techniques for virtual commissioning
Signal communication in robot cell using SIMIT templates
GUSTAV GUSTAFSSON, MARTIN GUSTAVSSON

Supervisor: ATLE ZVANTESSON, ÅF
Examiner: PETTER FALKMAN, Department of Electrical Engineering

Bachelor's Thesis 2019:06
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Digital twin of a robot cell in virtual reality.

Typeset in LaTeX
Gothenburg, Sweden 2019

Efficient modelling techniques for virtual commissioning
Signal communication in robot cell using SIMIT templates
GUSTAV GUSTAFSSON, MARTIN GUSTAVSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The development of virtual commissioning, a method for virtually simulating an assembly line or robot cell, has taken a big leap forward within the last couple of years. This because of the need for debugging Programmable Logic Controller (PLC) code and decreasing ramp-up time for planned robot cells. Due to continuous increase in computer performance, virtual commissioning (VC) has become a reliable and secure tool for verification of signal communication, robot movement and collision detection in a virtual environment. This Bachelor's Thesis has overhauled the signal communication between a robot arm and its surrounding virtual machines in an already created VC cell, and found a method for creating standardized communication templates from a program sequence list made by ÅF. By finding similarities in the communication in the digital twins (digital representations of physical machines), states or conditions have been found. The signals from the robot are used for setting these conditions and therefor executing the program of the cell in a correct order. Collecting tag names from a signal list and implementing them in the created state templates by checking condition order in the sequence list, machines could be created in an efficient and structured way. The method included auto-generated templates from a bulk engineering file, where signals were stated in a structured way for each kind of template needed for creating a specific machine, surrounding the robot in the cell. With Microsoft Excel, the bulk engineering file could be filled out entirely by creating code in the Visual Basic for Applications (VBA) editor. This VBA code template could be used for creating machines if a sequence and signal list of correct type were handed. The project recommendation for further work are to develop the code in VBA so that it can work with different types of sequence and signal lists. Also to expand the template library for machines that differs in signal type and structure.

Keywords: Virtual Commissioning, Auto-generated templates, SIMIT, State Machine, Bulk Engineering

Acknowledgements

First and foremost, we want to thank our supervisor Atle Zvantesson at ÅF that has been of great help throughout the project. He has continuously advised us regarding programming structure and functionality that he earlier has developed. Without his inputs and knowledge regarding the software used for programming the machines, the thesis would not have been accomplished. Bassam Massouh, who did his Master Thesis for the same virtual commissioning cell, was also of great support when software related issues was encountered.

We would also like to thank our examiner Petter Falkman and our supervisor Ludvig Ekström at Chalmers University of Technology. They both emphasized the importance of good pre-work with the project proposal and the planning report which came in useful. It lead to both greater understanding of the project and the value of time scheduling a project.

Last but not least we want to thank Andreas Buhlin at ÅF for letting us work with this thesis and the acceptance of scaling down a master thesis to a bachelor's.

All peers mentioned above is the reason for this thesis to have been made possible. We sincerely and deeply appreciate the assistance. Thank you.

Gothenburg, June 2019

Gustav Gustafsson and Martin Gustavsson

Contents

List of Figures

List of Tables

Abbreviations

BOOL Boolean (variable type)
Byte A group of 8 binary digits (variable type)
DES Discrete Event Simulation
I/O-list Input/Output-list
PLC Programmable Logic Controller
SIM Advanced Simulation Advanced
SIMIT Simulation platform for Virtual Commissioning
TIA Totally Integrated Automation
var Variable
VBA Visual Basic for Applications
VC Virtual Commissioning
VR Virtual Reality
.xls-file Excel spreadsheet-file
ÅF Ångpanneföreningen

1
Introduction

The opening chapter, Background, presents an introduction to the project. This is followed by the Purpose section which will explain why there is a need for an efficient method when creating communication in between digital twins and robot. The introduction will also provide aims and scope for the project.

1.1 Background

Virtual commissioning uses a virtual model of an existing or planned robot cell or production line connected directly to the control system to enable virtual testing and verification at an early stage in development. As for the existing robot cell, VC can enable improvements in efficiency by safe verification of signal communication in an virtual environment before implementing changes in the current program code[8]. Virtual commissioning also helps to shorten start-up and ramp-up time when commissioning[4]. The demand for Virtual Commissioning in the production industry is increasing hence to the economic advantages it can bring[6].

Machine vendors and competing platform providers are in a race to provide tools and software support for Virtual Commissioning. However Virtual Commissioning is still in an early stage, therefore there is a need to develop methods to match the demands. The vision is to be able to test Programmable logic controller (PLC) and Robot behaviour of an automated cell in a virtual environment without requiring any changes to the program code to make it compatible with the simulation model. There is hence a need to create digital twins (, which are virtual and digital representations of physical machines,) for all machines and equipment in the robot cell that interacts with the PLC and Robot.

Stress test the PLC and robot program in a virtual environment is made possible by the signal interface of all virtual objects in the robot cell. They should be created, as exact as possible when virtually commissioning a digital twin. For example, in how a honing machine and a welding tool communicates with the robot. However, efficiency in the making of signal interface for virtual objects is low due to time consuming programming. Similarities between how the objects are affected by the robot signals has been found and the aim of this thesis was to make the communication settings as efficient as possible.

1.2 Purpose

The need for efficient methods when creating digital twins in a virtual environment is the purpose of this thesis. When the demand of implementing new machines to an already working robot cell, efficiency is of great importance. The task of programming each individual machine is time consuming and by finding a method that solves this issue is of economical interest for both the customer and vendor providing the service. The method is also allowing the distributor with little insight in the machine structure, to use the tools developed for this thesis to create a requested machine.

1.3 Research questions

In what way can a modelling technique be created for signal communication in SIMIT, which is both easy to use and that saves the developer time to create the digital twin in a more efficient way?

In order to achieve a method worth implementing and using, the following problems was considered when developing the technique.

- Identifying the bottlenecks when creating digital twins.
- Identifying the problems of making previous digital twins.
- Look for possible shortcuts in the software used for creating the machine logic.
- Creating standardized templates to simplify the programming.
- Systemize the creation of connections between digital twins.

1.4 Scope

Digital twins are virtual objects programmed to behave in the same way as their physical equal. In this case, they have already been created by ÅF and was studied in order to find efficient ways of modelling digital twin communication. The PLC code, sequence list and signal interface list given and made by ÅF was first studied and the techniques that was found useful was applied in the making of new machines.

This thesis is limited to the usage of Siemens softwares such as PLC Sim Advanced, TIA-portal and SIMIT. The simulation and rendering software Robotstudio is a product of ABB but has not been included in the modeling technique hence to the developed method is only affecting the emulated connections in between digital twins in the virtual cell. The main work of the thesis covers the functions in SIMIT and the usage of importing from .xls-files.

The number of machines created was not the focus for this thesis. Finding an efficient technique that would thereby make it possible of creating machines in a time-saving way was of interest for this thesis.

2

Theory

The theoretical framework is to present the topics used for developing the templates and code in order to make an efficient modelling technique for virtual machines. This chapter will work as support to understand the modelling technique and how it was applied to the virtual commissioning cell.

2.1 PLC

Programmable Logic Controller (PLC) is a controller device equipped with a microprocessor that is used for programming a machine, robot or automation process. The PLC executes the program after reading input signals from the process. The logic then decides what outputs that are to be set true to the process. The signals that is sent to or meant to be sent from the PLC are called inputs and outputs. To get an overview of all signals that can be processed by the PLC, an I/O-list is used. The list supports information about size, number order and tag name of the signals. [5]

2.2 Digital twin

The copy of a physical machine or robot in a virtual environment is called a digital twin. The twins are made to emulate the behaviour of the machines and robot in the way they respond to signals and physical movement, this by adding the I/O interface for the physical machines to the twins. They are also supposed to have the same visual effects as their equal model by using 3D-animation softwares.[3]

2.3 Virtual commissioning

Virtual Commissioning is a method for testing a virtual copy of a robot cell by simulating the working process for the setup. The commissioning is built up by digital twins that are designed and programmed to replicate the behavior of the planned or existing physical robot cell.[12]
The digital twins is then connected to the PLC or by a softPLC in order to validate the exact signal communication and response of a real model.

2.4 Software for creating digital twins when virtual commissioning

The modelling technique presented in this thesis was programmed and structured by tools and software introduced in this chapter. This chapter will present how they are connected and why they have been relevant for the creation of the modelling technique.

There are four programs that need to work together to create and simulate the digital twin:

- TIA Portal
- PLC Sim advanced
- SIMIT
- Robot Studio

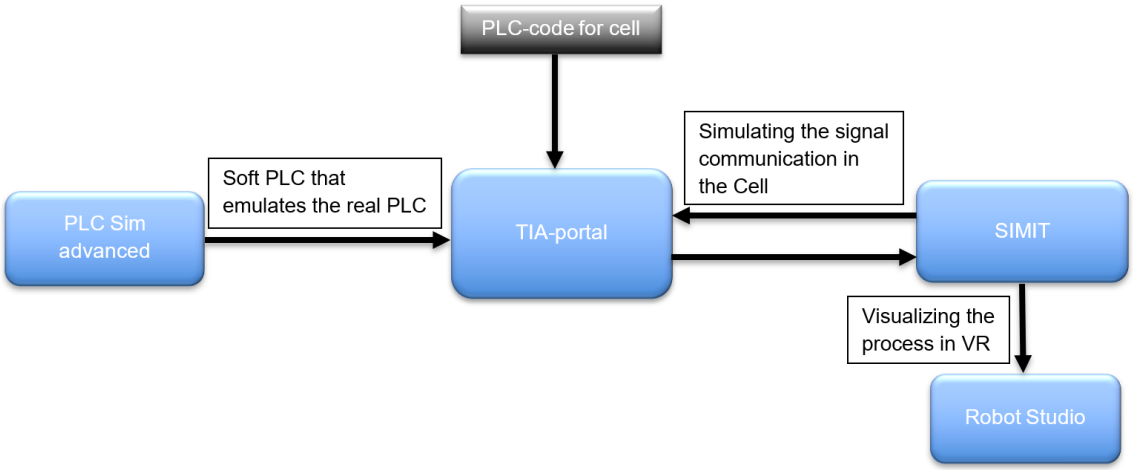


Figure 2.1: Visualization of how the programs communicate with each other

2.4.1 TIA portal

The TIA portal is an advanced engineering framework for implementing automation solutions. The software is used for configuration, programming, testing and diagnosis of the controllers in the Siemens product catalogue. PLC code is integrated with the software in order to use the functions for VC in the robot cell.[11]

2.4.2 PLC sim advanced

The simulation software PLC SIM Advanced is used for testing PLC code without the actual hardware. PLC SIM Advanced is a so called softPLC, a software that emulates the logic of a PLC hardware. PLC SIM advanced is connected to the

TIA-portal which holds the PLC code for the PLC in order to stress test the code in the software, to prevent the usage of stress testing physical models.[9]

2.4.3 SIMIT

In order to create the logic for the communication between the machines and robot the simulation platform SIMIT was used for this project. The templates created in SIMIT were developed to emulate the communication between robot and machines. By mimicking the machine communication, the virtual commissioning was able to complete the connections between digital twins.[10] SIMIT also communicates with Robot studio in order to visualize the process in VR.

2.4.4 Robot studio

To visually emulate the behavior of a real robot cell, robot studio is used to behave as the physical robot in its form and movement. The PLC logic is controlled in TIA portal and SIMIT controls the communication between PLC and robot studio. Robot studio works as the virtual illustration of the robot cell in movement and physical shape. The developer is able to use the software to analyze the movement and see if there are optimization possibilities to be made in the robot movement as well as to detect machine collisions in the cell.[7]

2.5 SIMIT templates

When using virtual commissioning, the connections in between the digital twins and PLC can be time consuming to create. In order to make the connections more structured and standardized, SIMIT templates has been created for this project. When making a connection for signals, they have to pass through function blocks that emulates the behaviour of the real signals of the machines and robot. SIMIT templates are using function block diagram programming that is a graphical language for presenting the function between input and output signals. In SIMIT, these so called function blocks is named components and is the core elements of the templates created for this thesis. [10]

2.6 Discrete-event simulation model

The structure meant to be implemented for the machines had to include several conditions that affects one another through templates. To simplify the structure the machine executes the program by processing different states in a certain order. It works as if one state is executed, then the program should continue with the next one. The behaviour could be seen as a Discrete Event Simulation (DES), were a condition has to be met in order to continue the sequence. [2]

In our case, the conditions that were to be fulfilled often included two signals or more to continue the program sequence. This was taken in consideration when the creation of state templates were made. The aim was to create state templates that were standardized and could be added to a template library were they could be stored for future creation of machine communication.

2.7 Visual Basic for Applications (VBA)

Visual Basic for Applications (VBA) is an implementation of the programming language Visual Basic created by Microsoft for Excel. The feature is useful for automating repetitive tasks or for customizing existing applications, included in Excel, to meet a user's personal needs[13]. The code can either be created in response to the user input which is translated into Basic code, specified for each control. The developer can also write code manually in the Visual Basic code editor which has been made for this project.

3
Methods & Implementations

The modeling method created is meant to be used by taking a sequence list of how the machines interacts in the robot cell and a signal interface list that provides the correct tag names for the signals sent to and from the machine. These two lists (figure 3.1) will enable generating code by finding conditions, states, in the sequence list and implement them using the correct tag names from the signal list. Templates was created for the cause of similarities found in the sequence of the machines. These templates are used to eliminate the repetition of the programming. To fill in the signals of the templates a so called bulk engineering file was created in Microsoft Excel (figure 3.10). This bulk engineering file is then imported to SIMIT which builds the program for the machine using the templates in the order the programmer has typed in signals in the bulk engineer file.

	A	B	C	D	E
1		**Load Work pice**			
2		Machine --->> Cell		Cell --->> Machine	
3		Auto	<<--->>	Auto	
4	Set	Load Work piece	--->>		
5			<<---	Article number	Set
6	Set	Article number_Machine_Setup	--->>		
7	Set	Machine ready for robot entrance	--->>		
8			<<---	Article number	Reset
9	Reset	Article number_Machine_Setup	--->>		
10			<<---	Robot Outside Machine	Reset
11			<<---	Ring Loaded	Set
12			<<---	Robot Outside Machine	Set
13	Reset	Load Work piece	--->>		
14	Reset	Machine ready for robot entrance	--->>		
15			<<---	Ring loaded	Reset

	A	B	C	D
3			**Signals from machine to cell**	
4				
5	**Address**	**Type**	**Function**	**Comment**
6			**General**	
7	0	DWORD	PNPN-Diag	0-3 Reserved for PN/PN couper diagnostics
8	4.0	BOOL	Auto	Machin in Auto
9	4.1	BOOL	Life bit	Life bit from machine
10	4.2	BOOL	Alarm	Alarm set in the machine.
11	4.3	BOOL	Close Cell Door Request	Cell Door needs to be closed for automatic mode in machine.
12	4.4	BOOL	Cell entrance OK	Cell Door can be opened. Machine is in safe parking position.
13	4.5	BOOL	RequestPutCapOn	Used to display event on the Cell HMI. Operator needs to go to the honing sluice
14	4.6	BOOL	RequestPutCapOff	Used to display event on the Cell HMI. Operator needs to go to the honing sluice
15	4.7	BOOL	Reserv47	Reserved
16			**Handshake**	
17	5.0	BOOL	Load workpiece	Machine in auto, cycle started
18	5.1	BOOL	Unload workpiece	Mission request to unload workpiece
19	5.2	BOOL	Machine ready for robot entrance	Machine has the right program loaded and the loading area is fully prepared for the robot entrence.
20	5.3	BOOL	Sequence Reseted Ack	The Machine has reseted the sequence used for cell handshake.
21	5.4	BOOL	Reserv54	Reserved
22	5.5	BOOL	Reserv55	Reserved
23	5.6	BOOL	Reserv56	Reserved
24	5.7	BOOL	Reserv57	Reserved

Figure 3.1: Part of a sequence list (left) and part of a signal list (right) for a machine in the cell

In total seven templates was created:

- The State Machine template (3.3)
- The Flip Flop template (3.4)
- The Bytes to double word template (3.5)
- The Double word to byte template (3.6)
- The Bytes to bits template (3.7)
- The Bits to byte template (3.8)
- The Compare template (3.9)

3.1 Identify problems when creating communication in SIMIT

The issues that often occur when implementing new machines had to be found by trying to create a machine from the beginning in order to understand the process. Hence to our little experience in creation of virtual machines an old model could be studied, supplied by ÅF.

One of the parts that slowed down the process of creating connections for the signals was the interface of the software. The dragging of wires in between the function blocks in SIMIT was not efficient in comparison of an auto-generated type of connection wiring that would be made possible by importing data from an excel file.

The other problem found was the inconsistency in the creation of connections in SIMIT. The communication between robot and machine was in some cases consistent by a small extent, except for the number of inputs for the conditions. One state could be dependent by one signals being set and another by two for example (figure 3.2). The creation of state templates had to rely on this variation in order to make an efficient and flexible method.

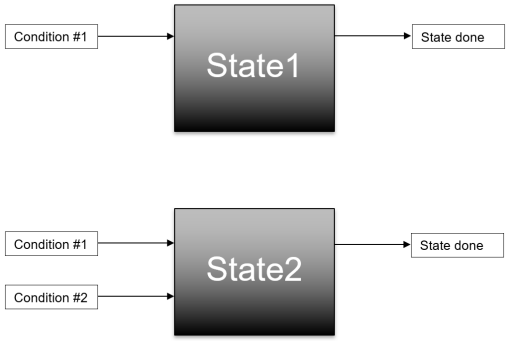


Figure 3.2: Depiction of how different states are dependent on different amounts of conditions

By studying the sequence list for a machine, a state order could be found by looking for conditions when the machine sent signals to the robot. This could then be implemented as a standardized state-template. The state-templates created from the sequential communication table (sequence list), had to ensure that not two states could be active all at once. The machine was not allowed to start doing things out of order and sending signals to the robot making it able to move, when it needed to wait for the machine to do something else first. For this problem a template was created which could take account of an active state and disable others from being activated at the same time. The most efficient way of making this possible was to create set and reset signals for the state levels. They then had to be connected to a template that took enabling and disabling of a state in consideration.

When auto-generating templates and the connections for signal communication, naming tags became an issue. The tag naming of the signals were in some cases named similar or the same to one another, despite the signals belonged to different machines. The problem occurred when a new machine was implemented and the tag names for already created machines in SIMIT provided the exact same name. This led to failure when compiling the connections. To solve this, a standardization when

naming tags was to be implemented. Firstly by naming the signals with the machine type it was used for, in the beginning of its tag name. This enabled sorting of similar signal names between machines and the standard could easily be implemented in the creation of VBA code for Excel. Secondly to separate the states by adding an order number to the end of the tag names allowed to control the activation of states.

To summarize identified problems when creating machine communication:

- Time spent on understanding the machines and the communication in between digital twins in the robot cell.
- The software interface, where a large amount of signals had to be connected by hand led to great inefficiency.
- An inconsistency when creating machines which led to difficulties when debugging signal communication in between machines.
- When implementing states and dividing the machine communication into conditions, the modelling technique had to ensure that there were no possibility whatsoever to activate several states at once.
- The naming of signals could lead to repetition and over-writing of already existing signal tag names.

How to solve the identified problems:

- Create a bulk engineering file to easy understand and structure the machine states and signals included.
- By using the function of importing .xls-files (Excel format), eliminate the problem of (by hand) dragging the connections in between signals in SIMIT.
- The problem of inconsistency in the creation of machine programming was reduced hence to the creation of standardized templates. These templates also solved the problem of having to connect signals by hand in SIMIT.
- To solve the problem of having the SIMIT program only run the states one at a time was solved by adding a flip flop template with flags that enabled a state or disabled it by either set or reset the flip flop that handled the executed state signal.
- A standard when naming the tags depending on machine type dealt with the problem of repetitive naming in between signal tags for the machines.

3.2 Literature review

The identified problems in the creation of signal connections in SIMIT, made it possible to search for previous studies written in the field of virtual commissioning and the creation of virtual machines.

As mentioned in the section for identifying problems when creating communication, states for machines are commonly used and one of the well-known methods are Discrete Event Simulation (DES). [2]

A method for recommissioning and virtual commissioning a changeable manufacturing system was found. Mortensen (ST Mortensen et. al. 2019) stated a method to be used when reusing virtual commissioning models. He firstly suggested to recognize reconfiguration complexity. Thereafter identify needed elementary reconfiguration ability when VC. For the last step he suggested to perform the actions indicated for the class VC in the table stated in the report. The table consists of four elementary abilities that can be performed. Rearranging, scaling, adding/removing or exchanging process modules. For VC , the suggestion was to modify interface of standard virtual devices. Standardizing virtual devices was shown to be a key element for reusing VC.[4]

Because of the many platform providers in the field it is hard to implement a method due to differences in software functionalities, for example the .xls-file import used for the method presented in this thesis. Therefor, as mentioned in the scope, the method could only be executed by Siemens software and no literature could be found on other efficient methods for creating communication in SIMIT .

Virtual commissioning has been around for awhile and has shown to be valuable for companies because of the time aspects. According to Reinhart [6] it was shown that virtual commissioning may lower the commissioning time with 75%. Mainly because the physical commissioning time usually is used to debug software.

Our mission with the project was to find an efficient modelling technique for virtual commissioning and by that we found a method for creating template libraries for code[1] that we were about to apply for the thesis. However we did not happen to find any use of code language templates more than the function blocks used for the templates. The only thing comparable with the code library would be the VBA coding in Microsoft Excel.

3.3 Creation of templates in SIMIT

In order to use the signals for the states, most of them had to be converted before they could be implemented in the program. The templates created had to include the parts were signals could be converted from both Byte to bits and vice versa. The most commonly used signal types was the setpoint for the creation of conversion templates, so that the signals could be used for the state machine template.

The sequence of the machine found in the sequence list had to be divided into states where a template were to be standardized and created with the respect to these states. The state machine template consists of one flag that informs that the state is to be set active. In order to apply such a function, another template for disable other states from being active had to be created. For this purpose, a flip flop had to be created for every implemented state in order to keep track of what state to be set or not.

3.3.1 The State machine template

The State Machine template (figure 3.3) is the most important template and it makes sure that the machine does everything it is supposed to do and in the correct order. The template keeps the signal b0 active until our conditions b1-b7 are fulfilled at which point the signal W0 tells the next state that it can start and also resets the signal b0.

The conditions in this template is based on different states taken from the sequence list and how they often tend to change conditions in the machine communication, as shown in figure 3.2.

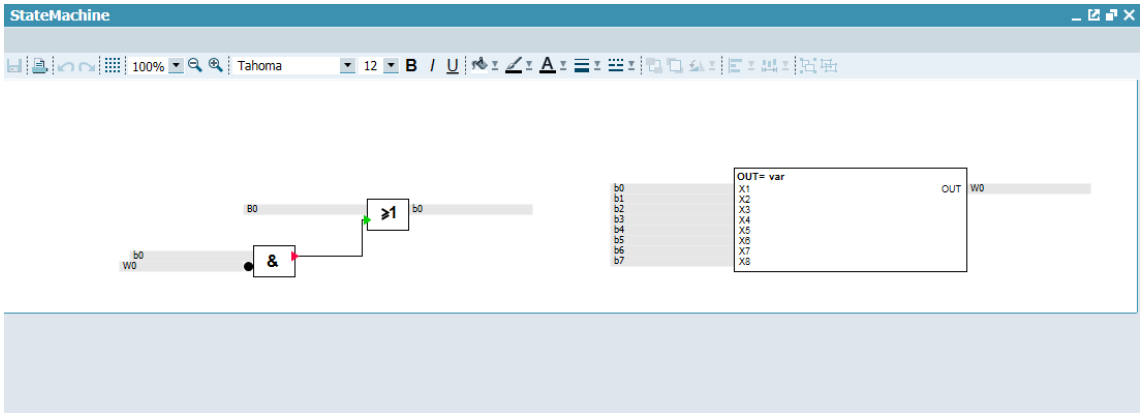


Figure 3.3: State machine template in SIMIT

Tag name:	**Usage:**	**Name examples:**
W0	State finished	MACHINENAME_state2_finished
B0	Previous state done	MACHINENAME_state1_finished
b0	What the machine should do in the state	MACHINENAME_set_Load_Workpice
b1-b7	Conditions that the machine needs before it can move on to the next state	MACHINENAME_robot_outside_machine
var	How many conditions needs to be true before the state is done	X1 AND X2 AND X3

Table 3.1: Signals in State machine template

For the B0 tag, a signal that indicates that the previous state has been executed is meant to be connected to this template input. If there are several states for the machine created, the B0 input is a way to let the program know that the previous state has been executed and therefore the following one can proceed.

The signal connected to b0 is the signal activated in this state. For example if the machine is to set the “Load workpiece” signal to TRUE, then this signal is put here. When the B0 signal tells the state that the previous state is done, the b0 signal is activated. The inverted W0 signal to the left in figure 3.3 holds the b0 signal set TRUE until the conditions of b1-b7 is activated. This then activates the W0 signal which sets the b0 signal to FALSE. Signals b1-b7 are conditions of when the robot has sent back the signals that the state demands in order to move to the next state. The state will, when conditions are met, set the W0 signal which deactivates the current state and also tells the next state that it is ready to start. This means that the W0 signal needs to be the same as the B0 signal for the next state. For example if we are in state2 then our W0 = "state2 done", then in state3 we should have B0 = "state2 done". This pattern is then repeated until the program is done at which point it returns to the first state and awaits to start again once called upon.

Since different states might have different numbers of conditions, as shown in figure 3.2, a max of 6 signals can be used as condition in this template (b1-b7). The amount of signals that is to be used in a state, is placed in the block with the X1-X8 inputs to the right in figure 3.3. As it says at the top of the block the output "OUT" of the block is determined by the "var". Since X1 is connected to our b0 signal which is the active state signal, X1 must always be in the "var" statement. Then if one condition occurs, "X1 AND X2" is written and if two conditions occurs, "X1 AND X2 AND X3" is written, and so on all the way up to the maximum om 6 conditions.

3.3.2 The Flip flop template

To be able to set and reset signals using other signals, a flip flop template had to be created. Because the handshake sequence between the machine and robot in the cell is sometime dependent on setting a signal and sometime resetting a signal, the signal had to be set even after the state was done in order to be able to reset it at a later time to trigger the handshake. For example in figure 3.20, the signal "load work piece" is to be set in the beginning of the sequence and then reset nearing the end of the sequence. In order for it to be kept TRUE even after the sequence has left the state that sets the signal, the flip flop template is used to ensure that the signal stay true. Then in the state were it needs to be reset, the reset signal connected to the flip flop is used. The flip flop template consists of the reset signal (b1) resets the output (W0) and the set signal (b0) that sets the output.

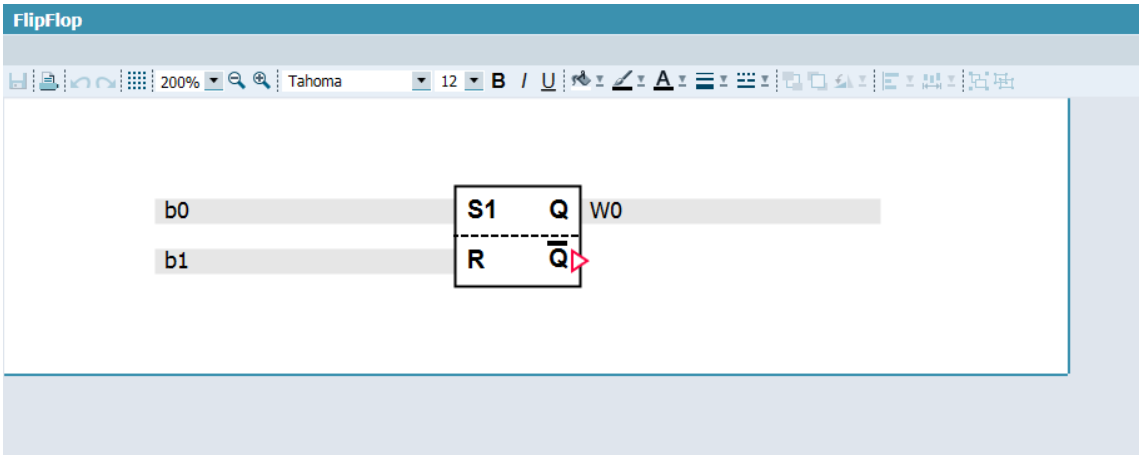


Figure 3.4: Flipflop template in SIMIT

Tag name:	**Usage:**	**Name examples:**
W0	Output signal	MACHINENAME_loadworkpiece
b0	The set signal for the output	MACHINENAME_set_loadworkpiece
b1	The reset signal for the output	MACHINENAME_reset_loadworkLoadpice

Table 3.2: Signals in State machine template

3.3.3 The bytes to double word template

Some of the signals were sent from the robot as bytes but were needed in word-format so we made a template to fix that. The bytes sent from the robot is used in the sections SOURCE B0-B3 and then transferred to a double word to used for the state machine.

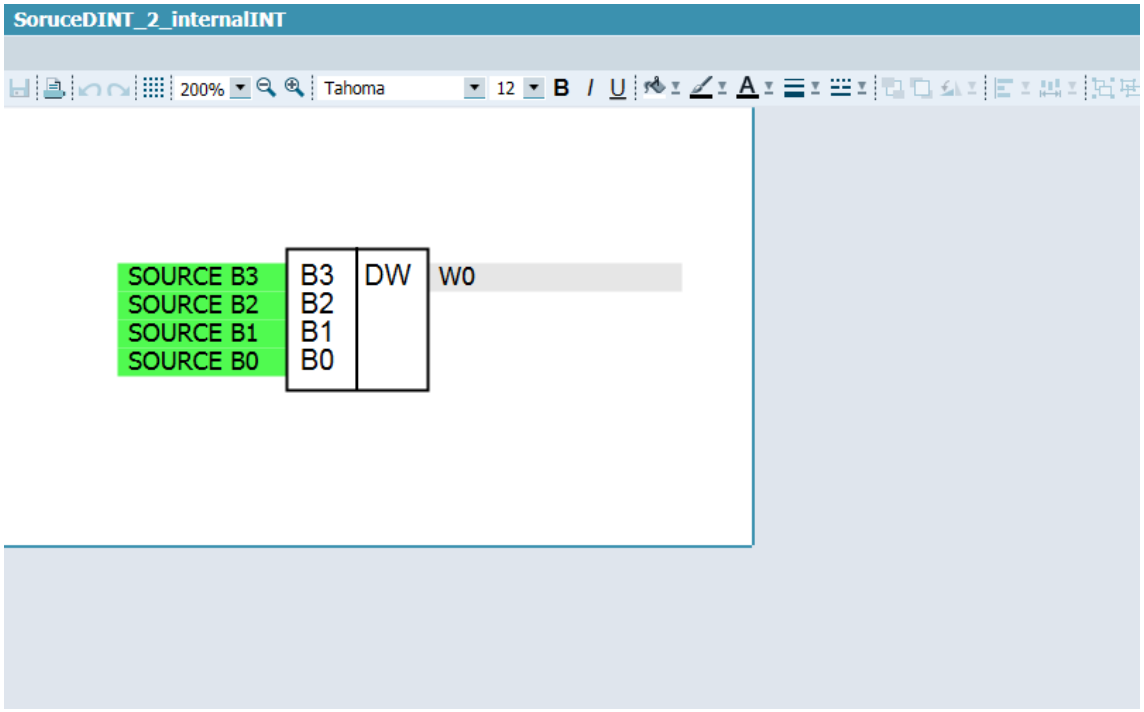


Figure 3.5: Byte to double word template

Tag name:	**Usage:**	**Name examples:**
W0	Output signal	MACHINENAME_articlenumber
SOURCE	The source of the signal	MACHINENAME_
B0-B3	The address of the bytes	MACHINENAME_HW_input

Table 3.3: Signals in Byte to double word template

3.3.4 The double word to bytes template

For sending the word signal back to the robot, the double word to byte template had to be created since to convert back the signal to it's original byte-format. The signals sent from the state machine template is converted through the template to the sections GOAL B0-B3, signals that are to be sent to the machine.

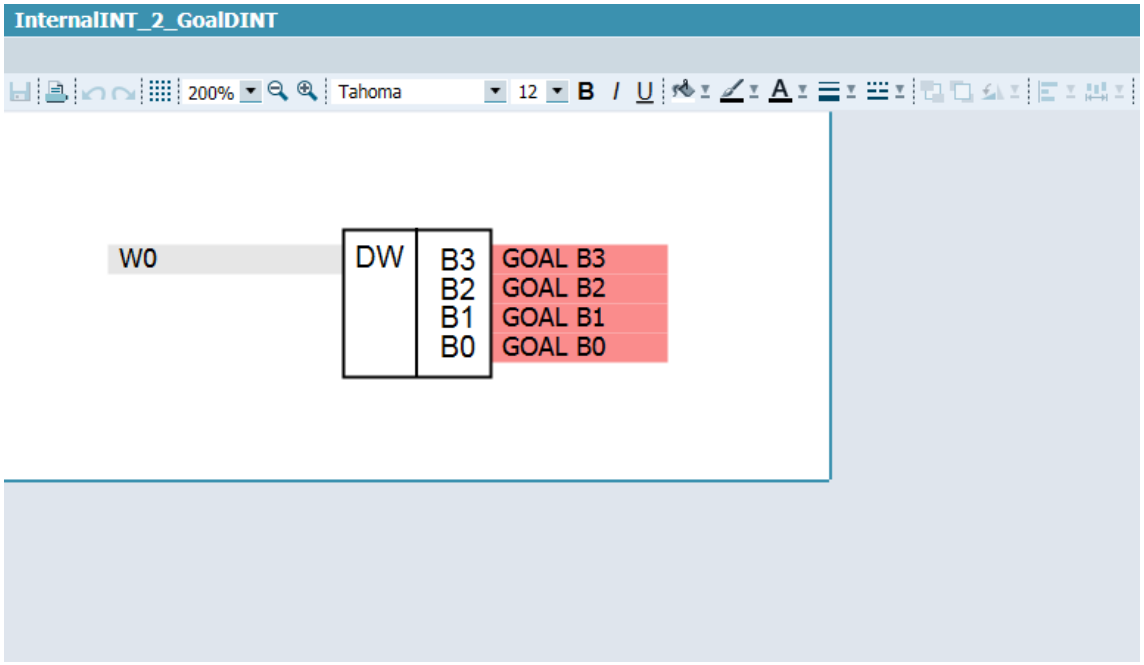


Figure 3.6: Double word to bytes template

Tag name:	**Usage:**	**Name examples:**
W0	input signal	MACHINENAME_articlenumber
GOAL	The goal destination of the signal	MACHINENAME_
B0-B3	The address of the bytes	MACHINENAME_HW_input

Table 3.4: Signals in double word to byte template

3.3.5 The byte to bits template

The robot sends most of the signals for our state machine as bytes, and since the individual bits from those bytes was needed to be able to use the signals properly in the state machine, a conversion of these bytes was made. These templates takes a byte sent from the robot and divides it into bits to be used in logical operations in our state machine.

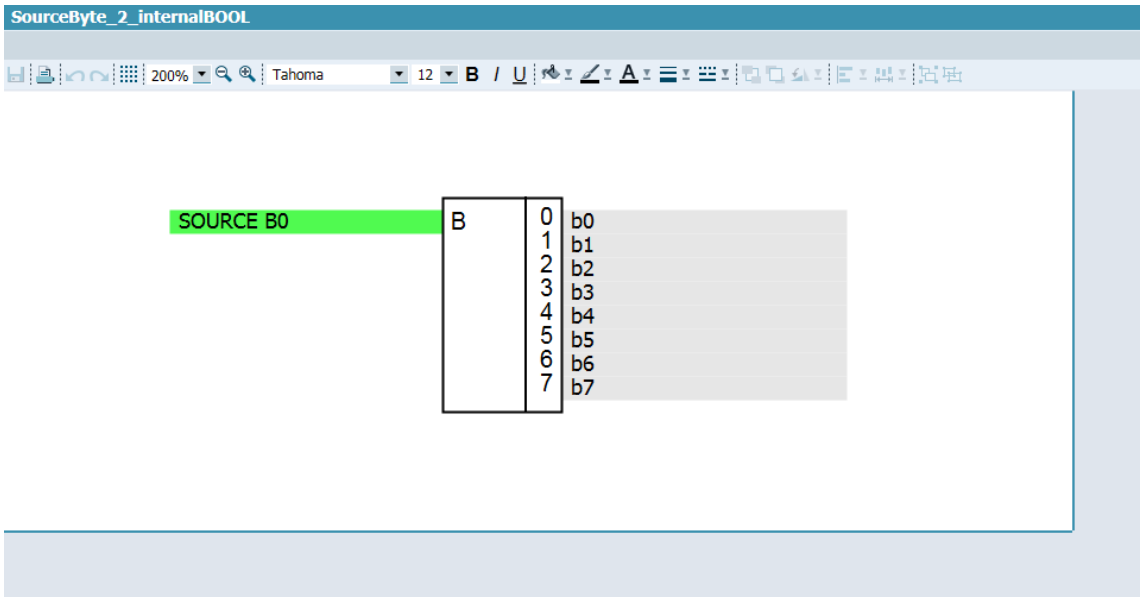


Figure 3.7: Byte to bits template

Tag name:	Usage:	Name examples:
B0	input signal	MACHINENAME_HW_something
SOURCE	The source of the signal	MACHINENAME_something
b0-b7	The internal bit signals	MACHINENAME_loadworkpiece

Table 3.5: Signals in byte to bit template

3.3.6 The bits to byte template

Just as the word to byte template mentioned before, converting bits back to bytes was also needed. The internal BOOL signals (bits) are converted to byte in this template so that they can be sent to the robot.

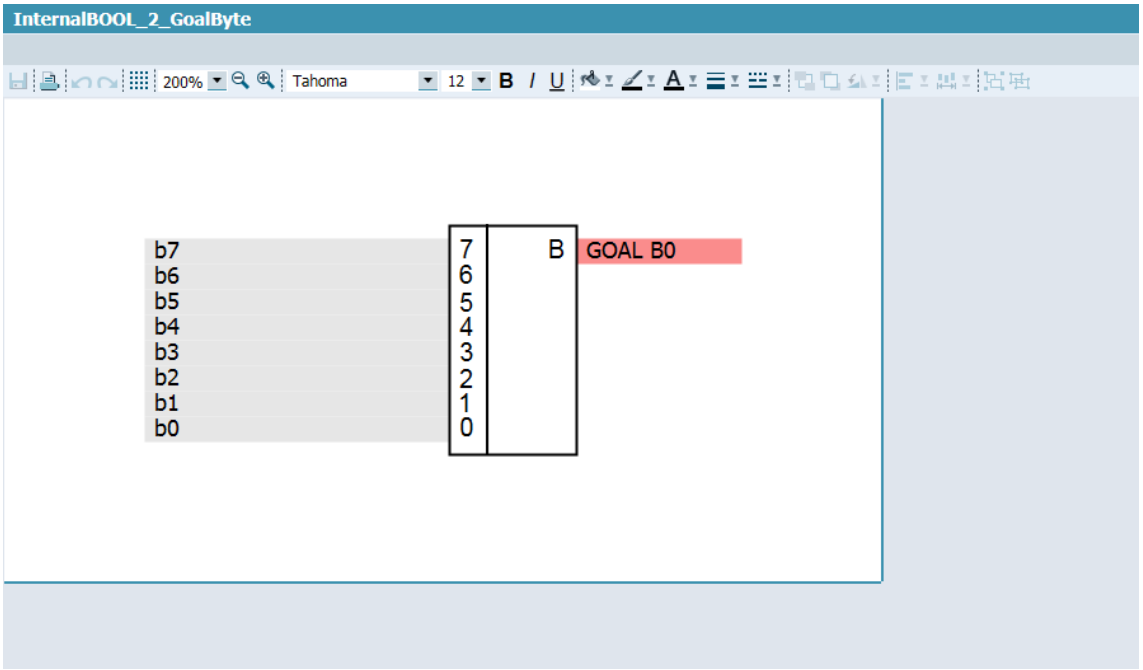


Figure 3.8: Bits to byte template

Tag name:	Usage:	Name examples:
B0	output signal	MACHINENAME_HW_
GOAL	The goal destination of the signal	MACHINENAME_
b0-b7	The internal bit signals	MACHINENAME_loadworkpiece

Table 3.6: Signals in bits to byte template

3.3.7 The Compare template

In order to see if the word signal coming from the robot had been changed, a compare template that could send a bool signal was needed. This template takes two signals, b0 and b1, and then it compares them and if the comparison is correct it sends a TRUE signal W0. Which type of comparison is going to be made is decided by the "VAR", e.g '<', '>' or '='.

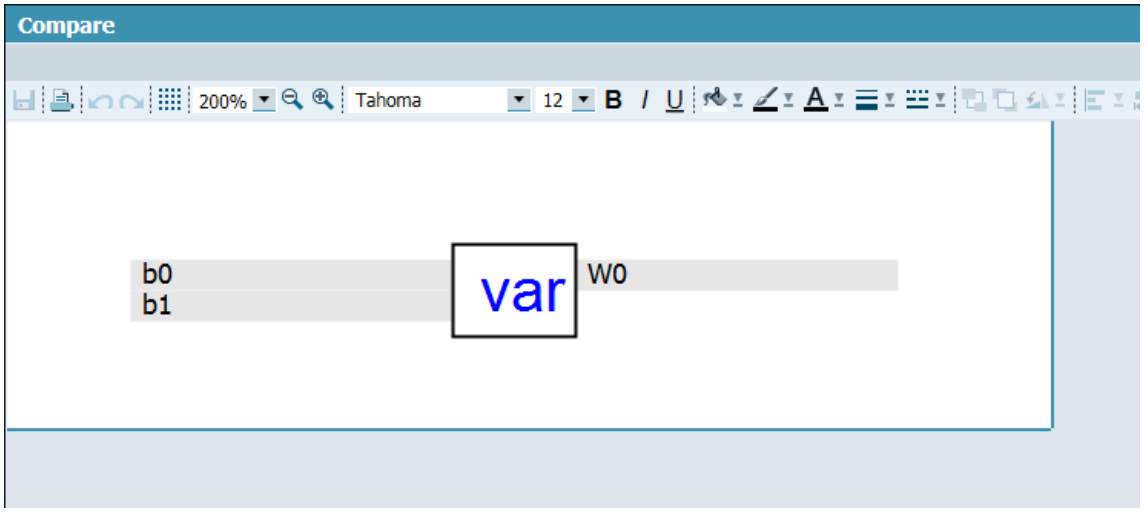


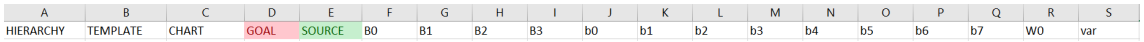
Figure 3.9: Compare template

Tag name:	Usage:	Name examples:
b0	singal to compare	MACHINENAME_articlenumber
b1	signal to compare	"any number"
Var	What type of comparison	<

Table 3.7: Signals in Compare template

3.4 Bulk engineering file

To be able to auto-generate templates for machine communication, structuring the creation of templates by typing each template per row in the bulk file with correct signals had to be implemented in order to keep track of templates and signals. The Bulk engineering file was an efficient way for structuring and auto-generating templates. The file consists of standardized columns in which indicates where to place signal connections to the templates in SIMIT.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3	b0	b1	b2	b3	b4	b5	b6	b7	W0	var

Figure 3.10: Bulk engineering file

The first column indicates in what hierarchy or machine to place the signals in SIMIT. It is then followed by what type of template that is to be implemented. Furthermore, in the following column, the chart that the program will be put is presented. The templates that has been created for the ÅF machines are the state machine template, the flip flop template and the conversion templates. Next section will explain the bulk engineering in more detail by explaining how to create all types of templates and where to insert what signals.

3.4.1 Creating state machine template in bulk engineering chart

When setting up the State machine template using the bulk engineering chart, the starting point was to check the sequence list to divide it into “states”. The idea is that the signal b0 is the signal for what action the state should do, for instance; set signal “Load Work piece”. Then check the robot to machine-side of the sequence list to see what the robot are supposed to do before moving on to the next action for the machine, these are called conditions and they are written in b1-b7. If there is only one condition then it is written in the b1 column and then in the “Var” column the operator writes: “X1 AND X2”. If the condition the machine is waiting for is a reset signal, the operator writes: “AND NOT” instead of AND in the “Var” column. At B0 the operator writes: “previous state done” so that the current state know that it should be activated. This signal just needs to be activated for a short time and then the state will stay active until all the conditions for the state have been fulfilled. Once the state is done the signal W0 is set which means that this signal should be “current state done”, which then is connected to the next state but then in the B0 column, which is “previous state done” but for the next state.

	A	B	C	D	E	F	G	H	I
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3
2	Test_Machine	StateMachine	Honing_Machine2	SM2	AS1	State6_done			

J	K	L	M	N	O	P	Q	R	S
b0	b1	b2	b3	b4	b5	b6	b7	W0	var
Honing_set_Load_workpiece	Honing_Article_number_bool							State1_done	X1 AND X2

Figure 3.11: Example of a State machine bulk engineer file with "AND" condition

	A	B	C	D	E	F	G	H	I
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3
2	Test_Machine	StateMachine	Honing_Machine2	SM2	AS1	State2_done			

J	K	L	M	N	O	P	Q	R	S
b0	b1	b2	b3	b4	b5	b6	b7	W0	var
Honing_set_Machine_ready_for_robot_entrance	Honing_Article_number_bool							State3_done	X1 AND NOT X2

Figure 3.12: Another State machine example but with "NOT AND" condition

3.4.2 Creating flip flop template in bulk engineering chart

The Flip flop template has only three signals: the output signal (W0), the set signal (b0) and the reset signal (b1). The signal W0 is designed to be set and reset at different states from the state machine. To make sure that the set signal doesn't reset once we leave the state, the flip flop is used. If the state want to set a signal the operator use "set_the signal that wants to be set" and vice versa "reset_the signal that wants to be reset".

	A	B	C	D	E	F	G	H	I
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3
2	Test_Machine	FlipFlop	Honing_Machine2	SM2	AS1				

J	K	L	M	N	O	P	Q	R	S
b0	b1	b2	b3	b4	b5	b6	b7	W0	var
Honing_set_Load_workpiece	Honing_reset_Load_workpiece							Honing_Load_workpiece	

Figure 3.13: Example of a FlipFlop bulk engineer file

3.4.3 Creating compare template in bulk engineering chart

In the compare template there are two signals that is in someway are being compared with each other. The b0 and b1 signal are the signals that are to be compared and if the comparison is correct then the signal W0 will be true. The VAR will determine what type of comparison will be done, so here the operator can type for example "<" (b0 less the b1), ">" (b0 greater than b1), "<>" (b0 not equal to b1) or "=" (b0 equal to b1) etc.

	A	B	C	D	E	F	G	H	I
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3
2	Test_Machine	Compare	Honing_Machine2	SM2	AS1				

J	K	L	M	N	O	P	Q	R	S
b0	b1	b2	b3	b4	b5	b6	b7	W0	var
Honing_Article_number	0							Honing_Article_number_bool	>

Figure 3.14: Example of a Compare bulk engineer file

3.4.4 Creating byte to internal Bool template in bulk engineering chart

The byte signals coming from the Robot to the machine needs to be converted into individual bits to use in the programming. In this template the byte is divided into 8 bool signals that is later used in the different templates. B0 is used for the byte signal and then b0-b7 is used for the bits.

	A	B	C	D	E	F	G	H	I
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3
2	Test_Machine	SourceByte_2_internalBOOL	Honing_Machine2	SM2	AS1	ROBOT_HW_IN_AS_BYTE.b04			

J	K	L	M	N	O	P	Q	R	S
b0	b1	b2	b3	b4	b5	b6	b7	W0	var
Honing_ring_loaded	Honing_ring_unloaded	Honing_reset_sequence	Reserv53	Reserv54	Reserv55	Reserv56	Reserv57		

Figure 3.15: Example of a Byte to internal bool bulk engineer file

3.4.5 Creating internal Bool to byte template in bulk engineering chart

This template is basically the same as the Byte to Bool but instead take the individual bit signals, coming from the machine that needs to be sent to the robot, and stitch them together as the complete byte. b7-b0 is again used for the bits and then W0 is used for the byte signal going through to the robot.

	A	B	C	D	E	F	G	H	I	J	K
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3	b0	b1
2	Test_Machine	InternalBOOL_2_GoalByte	Honing_Machine2	SM2	AS1	ROBOT_HW_OUT_AS_BYTE.b04				Honing_Auto	Honing_Life bit

L	M	N	O	P	Q	R	S
b2	b3	b4	b5	b6	b7	W0	var
Honing_Alarm	Honing_Close_Cell_Door_Request	Honing_Cell_entrance_OK	Honing_RequestPutCapOn	Honing_RequestPutCapOff	Reserv47		

Figure 3.16: Example of a bool to byte bulk engineer file

3.4.6 Creating byte to double word template in bulk engineering chart

In this template the byte signals coming from the robot are converted into a double word signal. B0-B4 is used for the byte signals that need to be converted and W0 is the double word output.

	A	B	C	D	E	F	G	H	I
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2	B3
2	Test_Machine	SourceDINT_2_internalINT	Honing_Machine2	SM2	AS1	ROBOT_HW_IN_AS_BYTE.b08			

J	K	L	M	N	O	P	Q	R	S
b0	b1	b2	b3	b4	b5	b6	b7	W0	var
								Honing_Article_number	

Figure 3.17: Example of a byte to double word bulk engineer file

3.4.7 Creating double word to byte template in bulk engineering chart

The double word signal needs to be converted back to byte before it can be sent back to the robot. B0-B4 are once again used for the bytes and the W0 signal is used for the double word.

	A	B	C	D	E	F	G	H
1	HIERARCHY	TEMPLATE	CHART	GOAL	SOURCE	B0	B1	B2
2	Test_Machine	InternalINT_2_GoalDINT	Honing_Machine2	SM2	AS1	ROBOT_HW_OUT_AS_BYTE.b08		

I	J	K	L	M	N	O	P	Q	R	S
B3	b0	b1	b2	b3	b4	b5	b6	b7	W0	var
									Honing_Article_number_Machine_Setup	

Figure 3.18: Example of a double word to byte bulk engineer file

3.5 Import signal list for machine types to bulk chart

When importing the signals for the machine, the operator checks the signal list (figure 3.19) that is provided and then see which type of signals it has and which of those are needed. If the operator for example have some byte signals but needs the individual bits for the logic in the machine, use the byte to bits template to convert the signal so that they can be used. Then fill out the bulk engineering excel file with the appropriate templates and signal names.

	A	B	C	D
3				
4			**Signals from machine to cell**	
5	**Address**	**Type**	**Function**	**Comment**
6			**General**	
7	0	DWORD	PNPN-Diag	0-3 Reserved for PN/PN couper diagnostics
8	4.0	BOOL	Auto	Machin in Auto
9	4.1	BOOL	Life bit	Life bit from machine
10	4.2	BOOL	Alarm	Alarm set in the machine.
11	4.3	BOOL	Close Cell Door Request	Cell Door needs to be closed for automatic mode in machine.
12	4.4	BOOL	Cell entrance OK	Cell Door can be opened. Machine is in safe parking position.
13	4.5	BOOL	RequestPutCapOn	Used to display event on the Cell HMI. Operator needs to go to the honing sluice
14	4.6	BOOL	RequestPutCapOff	Used to display event on the Cell HMI. Operator needs to go to the honing sluice
15	4.7	BOOL	Reserv47	Reserved
16			**Handshake**	
17	5.0	BOOL	Load workpiece	Machine in auto, cycle started
18	5.1	BOOL	Unload workpiece	Mission request to unload workpiece
19	5.2	BOOL	Machine ready for robot entrance	Machine has the right program loaded and the loading area is fully prepared for the robot entrence.
20	5.3	BOOL	Sequence Reseted Ack	The Machine has reseted the sequence used for cell handshake.
21	5.4	BOOL	Reserv54	Reserved
22	5.5	BOOL	Reserv55	Reserved
23	5.6	BOOL	Reserv56	Reserved
24	5.7	BOOL	Reserv57	Reserved

Figure 3.19: Signal list for a machine

3.6 Dividing the sequence list into states for machine communication

Most cells use so called "handshake" communication between the robot and the machines. This means that the robot does something and sends a signal to the machine telling it what it did and the machine then responds in the same way by first doing something and sending a signal back to the robot telling it what it did and then it continues like that back and forth. This is where the State machine template comes in to play and we divide our handshakes into states.

The best way to divide the sequence list into states is to first look at the machine-to-cell side of the sequence list. Use figure 3.20 as an example and in this list "set" means that the a signal is set to TRUE and "reset" means that it is set to FALSE. The machine starts of by setting the signal for load workpiece true. This is what's called the active state signal, in other words what the machine is doing in State1. After that on the cell-to-machine side of the sequence list there is the signal that the robot should respond with to the load work piece signal from the machine. This becomes the condition for State1, which means that the state is waiting for the conditions signals to be correct before it can move on to the next state. Once the

robot have responded with the proper signals it can move on to State2. Back to the machine-to-cell side of the sequence list and in state2 the machine is setting the "article_number_machine_setup" signal. After that, on the cell-to-machine side of the sequence list, the machine don't need any response from the robot to move on to the next state, which just means that it don't have any conditions in state2 to move on to state3. In state3 the "Machine ready for robot entrance" is the active state signal and in state3 the machine is instead waiting for the "article number" signal to be reset from the robot. State4 then resets the "article_number_machine_setup" signal that is set to true in state2, but now there are three different conditions that needs to be fulfilled before the move to state5.

This procedure is then repeated until the sequence list is done. In some cases the condition can be an integer in which case the usage of the compare template is needed to convert the signal to a bool. For example if the article number is suppose to be zero, is a condition the compare template check when the article number is zero and then sends the bool back as a condition to the state machine template.

	A	B	C	D	E
1		**Load Work pice**			
2		Machine --->> Cell		Cell --->> Machine	
3		Auto	<<--->>	Auto	
4	Set	Load Work piece	--->>		
5			<<---	Article number	Set
6	Set	Article number_Machine_Setup	--->>		
7	Set	Machine ready for robot entrance	--->>		
8			<<---	Article number	Reset
9	Reset	Article number_Machine_Setup	--->>		
10			<<---	Robot Outside Machine	Reset
11			<<---	Ring Loaded	Set
12			<<---	Robot Outside Machine	Set
13	Reset	Load Work piece	--->>		
14	Reset	Machine ready for robot entrance	--->>		
15			<<---	Ring loaded	Reset

Figure 3.20: Example of a sequence list for a machine

3.7 Auto-generating code

For the auto-generating part, it all comes down to what file you have to start with. If the developer are handed or has created the signal list and the sequence list that follows the same structure explained in the previous section. One could use the bulk engineering manual to type in signals by hand, but to auto-generate this step, VBA code could be used to transfer the idea of exporting signals and sequence list to the bulk engineering file. This step will be explained in the chapter VBA Excel for creating bulk engineering file .

Signals list → (VBA) → Bulk engineering file
Sequence list → (VBA) → Bulk engineering file

3.8 VBA Excel for creating templates in bulk engineering chart

For the auto-generating of signal communication, the way to structure it was to create standardized templates. Signals and states were then to be structured in the bulk engineering excel file in order to then export it and be implemented in SIMIT. Structuring the states by finding out what signals caused the conditions for the machine to be set and keep on with the program was made possible by the signal and sequence list. They were studied and from them the order of sorting the states, and where to connect and convert used signals was made.

The goal of auto-generating complete machines from the lists mentioned before, came from the idea of making an efficient modelling technique. To be able to import signals and states automatically to the bulk engineering file from the lists, code had to be used. Microsoft's Excel supports its own event-based programming implementation Visual Basic for Applications (VBA). By studying scripts used for find and replace names, characters and symbols it was made possible to sort out signals and name them in a standardized way.

3.8.1 Exporting signals list to the bulk engineering chart using VBA

For the signal list, exporting and converting signals was made so that they supported the naming standard and that they got connected to the right conversion type in the bulk engineering file. The code used for this implementation is presented in the Appendix on the rows 28-66.

As explained in the section "Import signal list for machine types to bulk chart", the code uses the same strategy as the operator would have. It copies the names of the signals from the signals list in the machine communication file. If the signal is of a certain type, the code ads the correct conversion template name in the template column in the bulk engineering file.

3.8.2 Create states from the sequence list to the bulk engineering chart using VBA

The sequence list were used for creating states and the flip flop templates taken in account and kept track of what states that were to be activated. The templates that has been created in SIMIT for this purpose, the state machine and the flip flops, contained both as mentioned in the chapter "Creating state machine template in bulk engineering file", a variety of signals and created states. When a condition was met in the sequence list, a new state had to be created in order for another condition to be met.

The code created the flip flops by identify all of the signals that activated the states conditions. Which in the sequence list was all of the signals in the first column, the column that presented the signals that was sent from the machine to the cell. The VBA code is presented in the Appendix on the rows 67-141.

4
Results

This chapter presents the results in this thesis.

4.1 Function Block Diagram templates

The templates could be applied in SIMIT and the signal templates emulated the program sequence of the robot cell as before. Our goal was to build a library with the most useful templates that we could think of using the machines from a previous robot cell. The idea is that this library can be further developed once new machines are needed to be built that can't take advantage of any of the already existing templates. In that case a new one is created to serve the purpose of it's machine and is then added to the library. In some cases the template might end up only being used for that particular machine and never used again, but this is still more effective than having to redo the same thing over again in SIMIT, sience it is very time consuming to program in SIMIT by hand.

This means that if a new machine is going to be programmed in SIMIT for a cell, the operator just checks the library to find the needed templates to complete the build for that machine, and if the machine can't be finished using only the templates in the library, a template is created for the function that operator needs and adds it to the library. The goal is that you should do as little work programming in SIMIT as possible and just pick out the parts you need from the library, never having to even program anything in SIMIT, you just fill out the Bulk engineering files and let SIMIT do the rest.

4.2 Bulk engineering template

The bulk engineering file could create templates in SIMIT according to the description explained in chapter bulk engineering file. The templates emulated the signal communication correctly after the auto-generating from bulk file to SIMIT. This was tested by adding buttons in SIMIT that simulated activation of robot-to-machine signals to find out if the sequence of the cell was emulated in the same way as for the sequence list. SIMIT also placed the templates in the order of how the bulk file was filled out.

If the demand for creating more machines, one can simply follow the requested machine's sequence list that explains of how the signals triggers the sequence and

fill out correct templates with associated signals in the bulk engineering file for that machine. The new bulk file will then auto-generate the requested templates in SIMIT.

4.3 VBA Auto-generating

The result of auto-generating the bulk file could be made based on the code described in the chapter “VBA Excel for creating templates in the bulk engineering chart”. The part auto-generated was the export of signals in the signals list to the conversion templates in the bulk file and the signals sent to the robot, used in the flip flop. Hence to limited time, the other templates such as the state template have to be filled out manually in the bulk file.

5
Discussion

The topics of the results, which are the templates, bulk engineering file and VBA code are the basis of the method developed for this thesis and will therefor be discussed.

The result showed that the method for implementing a more efficient method for signal communication in a VC cell were accomplished to some extent. The template library is consisted of usable templates when it comes to a VC cell using communication in between robot and machines. Still it has to be included more templates in order to work with other machines. Functions that was included in the machine we created the method for may differ to other machines, but the main components for creating a state based communication in a VC cell has been accomplished. The library created is a way of collecting templates in a structured way where everything can be reusable in other machines. Due to the lack of time, other machines was not tried to be created but the signal and sequence list for these machines were created and structured in a similar way to the one machine the method was based on, which leads us of thinking that the method created can be used for other machines as well.

The bulk file was an important tool for creating an efficient method. It eliminated one of the most time-consuming tasks of the creation of machines, which certainly was the connecting of signals to the templates in SIMIT. The method of using the .xls-file import in SIMIT was an efficient way of creating machines and the bulk file is an organized tool for debugging connections and signals for the templates if compiling errors occur. The naming of the templates has improvements to be made in order to understand what their purpose is for the machine. The ambitions of the templates are that an operator can read about the templates functionalities in notations, in either a manual for the library or creating comments in the bulk file.

If the method is used for a real case then the most efficient way of creating machines would be to demand the customer to deliver a signal and sequence list in order to understand the communication and what signals triggering the conditions for the machine. But most importantly to be able to auto-generate the whole process of creating machine communication. If the customer is using the structure of the lists handed by ÅF, then the VBA code could simply extract the signal list and place them in the correct order in the bulk file. All the operator has to do then is to use the sequence list and type in the right signals for the state machine template by hand. This, due to lack of time, was not made in VBA but can be implemented in code by following the typing sequence found in the method section of this paper.

Due to limited knowledge in VBA code this was the topic spent most time troubleshooting at. The code could be created by searching the web for functions possible in VBA and use them where we found the use for them. Microsoft Excel was a powerful tool and the ambition was to auto-generate the whole process. If more time was given and more functions were gained when collecting code for VBA, then the whole process would have been automated for the creation of signal communication in SIMIT. Other code languages could be of use for such a task but as a consequence of our limited time, VBA was a moderate language to learn and it was an integrated development environment for Excel. The VBA code could also be improved by commenting essential parts of the code where the operator could custom code for lists created in a different way from the lists handed by ÅF.

6
Conclusion & Future work

The beginning of the report presented one research question followed by five problems in which the project was set to answer. Here follows a conclusion for these five and how future work for this project can be developed.

- Identifying the bottlenecks when creating digital twins.
- Identifying the problems of making previous digital twins.
- Look for possible shortcuts in the software used for creating the machine logic.
- Creating standardized templates to simplify the programming.
- Systemize the creation of connections between digital twins.

The chapter of identifying problems when creating communication in SIMIT explains of the software interface, where the dragging wire-application being an inefficient way for creating connections between components. This was solved due to the xls-file import feature in SIMIT. The bulk engineering file made in Microsoft Excel was used for this cause. Both to eliminate the repetitive tasks and to structure templates, easier for debugging connections when templates being created in SIMIT. The research of identify previous problems when creating digital twins, looking for shortcuts in software and creating standardized templates have all been answered. Important to state is that future work can be made such as finding ways of expanding the template library that handles the communication in SIMIT by simply finding new machines. Machines that differs in behaviour from previous ones will probably have to include new templates, that then can be included in the library.

For the systemizing of connections and identify bottlenecks when creating digital twins can both be developed for future work due to time limits for this project. This thesis evaluated improvements in signal communication in SIMIT and some of the bottlenecks occurred when creating digital twins, was mostly found when connecting all softwares for testing the virtual commissioning cell. PLC SIM Advanced and Robot studio was software not used for developing the method of creating standardized templates, but used for virtually test that PLC and the virtual environment worked as for the previously developed program. Future work could include methods in how to connect software in a more efficient way.

The VBA code implemented for creating machines from the machine lists to the bulk file has improvements to be made. The code can be more compromised and well structured. The code also lacks the ability to adapt to different type of machine lists. The one code implemented for the machine used for this thesis will possibly

not be working for other machine list. This hence to the VBA code being more of a code template, where the developer can type in the size of the field the code is to look for signals in, rather than adaptative code. Future work can be made in making code suitable for different kind of lists to enable this step to also be auto-generated. Also to write code generating the fill out of the remaining state template is work left to be done. This will enable a complete auto-generation of machine communication from a sequence and signal list to connected templates.

Auto-generating repetitive steps thus to more efficient methods are evolved is always of great interest in the field of virtual commissioning.

Bibliography

[1] Yuli Hua Andreas Lu and Yuli Hua. "Software maintenance for Discrete-Event Simulation Models". In: (2018). URL: http://studentarbeten.chalmers.se/publication/255393-software-maintenance-for-discrete-event-simulation-models.

[2] Jerry Banks. "Discrete Event Simulation". In: *Encyclopedia of Information Systems* (Jan. 2003), pp. 663–671. DOI: 10.1016/B0-12-227240-4/00045-9. URL: https://www.sciencedirect.com/science/article/pii/B0122272404000459.

[3] *Digital Twin | Siemens*. URL: https://www.plm.automation.siemens.com/global/en/our-story/glossary/digital-twin/24465.

[4] "Operational Classification and Method for Reconfiguration & Recommissioning of Changeable Manufacturing Systems on System Level". In: *Procedia Manufacturing* 28 (Jan. 2019), pp. 90–95. ISSN: 2351-9789. DOI: 10.1016/J.PROMFG.2018.12.015. URL: https://www-sciencedirect-com.proxy.lib.chalmers.se/science/article/pii/S235197891831357X.

[5] *Programmable Logic Controller Introduction to Industrial Control Sys-tems and Operations*. Tech. rep.

[6] Gunther Reinhart et al. "Economic application of virtual commissioning to mechatronic production systems". In: *Production Engineering* 1.4 (Jan. 2007), pp. 371–379. ISSN: 0944-6524. DOI: 10.1007/s11740-007-0066-0. URL: https://www.plm.automation.siemens.com/global/en/our-story/glossary/digital-twin/24465%20https://support.industry.siemens.com/cs/ww/en/view/77362399%20https://community.plm.automation.siemens.com/t5/Tecnomatix-News/Virtual-Commissioning-A-practical-guide/ba-p.

[7] *RobotStudio - ABB Robotics*. URL: https://new.abb.com/products/robotics/robotstudio.

[8] *Siemens Integrates Virtual Planning with Physical Production to Maximize Manufacturing Productivity*. URL: https://www.plm.automation.siemens.com/global/en/our-story/newsroom/siemens-press-release/43428.

[9] "SIMATIC S7 PLCSIM Advanced - TIA Portal - Siemens". In: (). URL: https://w3.siemens.com/mcms/automation-software/en/tia-portal-software/step7-tia-portal/simatic-step7-options/s7-plcsim-advanced/pages/default.aspx.

[10] *SIMIT Simulation V9.0 Getting Started*. Tech. rep. URL: https://support.industry.siemens.com/cs/ww/en/view/77362399.

[11] *Totally Integrated Automation Portal One integrated engineering framework for all automation tasks. siemens.com/tia-portal TIA Portal-the new version.*

Tech. rep. URL: https://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_tia_portal_en.pdf.

[12] *Virtual Commissioning*. URL: https://www.plm.automation.siemens.com/global/en/products/tecnomatix/virtual-commissioning.html.

[13] *Visual Basic | Encyclopedia.com*. URL: https://www.encyclopedia.com/science-and-technology/computers-and-electrical-engineering/computers-and-computing/visual-basic.

A

Appendix 1

A.1 VBA code

```
1
2  Sub PasteSignals()
3
4  Dim x As Long
5  Dim y As Long
6  Dim Row As Long
7
8
9  Dim CopyWS As Worksheet
10 Dim TargetWS As Worksheet
11 Dim CopyWS2 As Worksheet
12
13
14 Set CopyWS = Workbooks("Honing_test.xlsx").Sheets(
15 "Signal Suggestion from AF")
16
17 Set TargetWS = Workbooks("BULKISEN_ver_1.xlsx").Sheets
18 ("Bulk engineer")
19
20 Set CopyWS2 = Workbooks("Honing_test.xlsx").Sheets
21 ("Sequence Suggestions from AF ")
22
23
24 'The row number to start from
25
26 x = 8
27
28 '_____________________________________________
29 'Loop below copies and pastes boolean signals
30 'from the signal suggestion list
31
32 For adr = 2 To 6
33
34     If CopyWS.Cells(x, 2) = "BOOL" Then
```

```
35          TargetWS.Cells(adr, 2) = "SourceByte_2_InternalBOOL"
36              For y = 10 To 17
37                  CopyWS.Cells(x, 3).Copy
38                  TargetWS.Cells(adr, y).PasteSpecial
39                  TargetWS.Cells(adr, y).Value
40                  = "Honing_" & TargetWS.Cells(adr, y).Value
41
42                  x = x + 1
43              Next y
44
45  '-------------------------------------------------------
46  'Loop below copies and pastes array signals
47  'from the signal suggestion list
48
49
50      ElseIf CopyWS.Cells(x, 2) = "Array of CHAR [0..17]" Then
51
52          TargetWS.Cells(adr, 2) = "SourceDINT_2_internalINT"
53
54          y = 10
55          CopyWS.Cells(x, 3).Copy
56          TargetWS.Cells(adr, y).PasteSpecial
57
58          TargetWS.Cells(adr, y).Value = "Honing_" &
59          TargetWS.Cells(adr, y).Value
60
61      End If
62
63
64      y = 10
65      x = x + 1
66  Next adr
67  '-------------------------------------------------------------
68  'Loop for creating flipflops, ads set and reset variables
69
70  Row = adr
71
72  For x = 4 To 28
73
74   y = 10
75
76      If CopyWS2.Cells(x, 1) = "Set" Then
77
78          TargetWS.Cells(Row, 2) = "FlipFlop"
79          CopyWS2.Cells(x, 2).Copy
80          TargetWS.Cells(Row, y).PasteSpecial
```

```
81          TargetWS.Cells(Row, y).Value
82          = "Honing_Set_" & TargetWS.Cells(Row, y).Value
83
84          y = y + 1
85          TargetWS.Cells(Row, y).PasteSpecial
86          TargetWS.Cells(Row, y).Value
87          = "Honing_Reset_" & TargetWS.Cells(Row, y).Value
88
89          y = y + 7
90          TargetWS.Cells(Row, y).PasteSpecial
91          TargetWS.Cells(Row, y).Value
92          = "Honing_" & TargetWS.Cells(Row, y).Value
93          Row = Row + 1
94
95      End If
96
97  Next x
98  '_____________________________________________________________
99  'State Machine Loop
100 x = 4
101 y = 2
102 Col = 11
103 If CopyWS2.Cells(x, y) = "" Then
104
105     y = y + 2
106     CopyWS2.Cells(x, y).Copy
107     TargetWS.Cells(Row, Col).PasteSpecial
108     TargetWS.Cells(Row, Col).Value = "Honing_" &
109     TargetWS.Cells(Row, Col).Value
110
111     x = x + 1
112     Col = Col + 1
113
114 Do While CopyWS2.Cells(x, y) <> ""
115
116     CopyWS2.Cells(x, y).Copy
117     TargetWS.Cells(Row, Col).PasteSpecial
118     TargetWS.Cells(Row, Col).Value = "Honing_" &
119     TargetWS.Cells(Row, Col).Value
120     Col = Col + 1
121     x = x + 1
122     Loop
123
124 y = y - 2
125 Else
126     CopyWS2.Cells(x, y).Copy
```

```
127         TargetWS.Cells(Row, 10).PasteSpecial
128         TargetWS.Cells(Row, 10).Value = "Honing_" &
129         TargetWS.Cells(Row, 10).Value
130         x = x + 1
131
132
133 '-------------------------------------------------------------
134 'Replaces all spaces with underscores
135
136 Workbooks("BULKISEN_ver_1.xlsx").Sheets("Bulk engineer")
137 .Columns("A:S").Replace _
138  What:=" ", Replacement:="_", _
139  SearchOrder:=xlByColumns, MatchCase:=True
140
141 End Sub
```