YRKESHÖGSKOLAN NOVIA

Development of Control Panel Cost Calculation Tool

Tony Bäck

Bachelor's Thesis

Electrical Engineering

Vasa 2017

BACHELOR'S THESIS

Author: Tony Bäck
Degree Programme: Electrical Engineering, Vaasa
Specialization: Electrical Power Engineering
Supervisors: Ronnie Sundsten, Mathias Sjöberg

Title: *Development of Control Panel Cost Calculation Tool*

Date April 12, 2017 Number of pages 26

Abstract

This Bachelor's thesis consists of a cost calculation tool for control panels to power plants. The cost calculation tool shows material costs and makes needed material lists for the control panels. Commissioner for the Bachelor's thesis is ABB Power Generation.

The objective for this thesis is to, in an easy and effective way make material lists with basic and option prices specified. It should also be possible to compare the final bill of material from the design tool against the material from the cost calculation tool. The cost calculation tool will be used in design of the control panels to inform the control panel manufacturer about needed material and price.

The result is a cost calculation tool that is made in Microsoft Excel using macro programming.

Language: English Key words: Microsoft Excel, Modular Power Plants, Cost

EXAMENSARBETE

Författare: Tony Bäck
Utbildning och ort: Elektroteknik, Vasa
Inriktningsalternativ: Elkraftsteknik
Handledare: Ronnie Sundsten, Mathias Sjöberg

Titel: *Utveckling av ett kostnadskalkylprogram för kontrollpaneler*

Datum 12 april 2017 Sidantal 26

Abstrakt

Detta examensarbete omfattar ett kostnadskalkylprogram för kontrollpaneler till kraftverk. Kostnadskalkylprogrammet visar behövligt material och materialkostnader för kontrollpanelerna. Uppdragsgivare för examensarbetet är ABB Power Generation.

Målet med examensarbetet är att man på ett enkelt och effektivt sätt ska kunna göra materiallistor med standard och tillvalspriser specificerade. Det skall också vara möjligt att jämföra den slutliga apparatlistan mot materiallistan som kostnadskalkylprogrammet genererar. Kalkylprogrammet kommer att användas vid design av kontrollpanelerna för att informera kontrollpaneltillverkaren om behövligt material och pris.

Resultatet blev ett kostnadskalkylprogram som är utfört i Microsoft Excel med hjälp av makroprogrammering.

Språk: engelska Nyckelord: Microsoft Excel, modulkraftverk, kostnad

OPINNÄYTETYÖ

Tekijä: Tony Bäck

Koulutus ja paikkakunta: Sähkötekniikka, Vaasa

Suuntautumisvaihtoehto: Sähkövoimatekniikka

Ohjaajat: Ronnie Sundsten, Mathias Sjöberg

Nimike: *Ohjauspaneelien kustannuslaskentaohjelman kehittäminen*

Päivämäärä 12 huhtikuuta 2017 Sivumäärä 26

Tiivistelmä

Tämä opinnäytetyö käsittää laskentaohjelman voimalaitosten ohjauspaneelien kustannuslaskentaa varten. Laskentaohjelma näyttää ohjauspaneelien materiaalikustannukset ja listaa tarvittavat materiaalit. Opinnäytetyö on tehty ABB:lle Power Generation-yksikölle.

Opinnäytetyön tarkoitus on saada helppo ja tehokas tapa tuottaa materiaalilistoja missä perus- ja lisävarustehinnat on eriteltynä. Pitää myös olla mahdollista verrata suunnitteluohjelman lopullista materiaalilistaa laskentaohjelman materiaalilistaan. Laskentaohjelmaa tullaan käyttämään ohjauspaneelien suunnittelussa siten, että ilmoitetaan ohjauspaneelien valmistajalle tarvittavat materiaalit sekä hinta.

Tulos on laskentaohjelma, joka on tehty Microsoft Excelissä käyttämällä makro-ohjelmointia.

Kieli: englanti Avainsanat: Microsoft Excel, moduulivoimalaitos, kustannus

Table of contents

1 Introduction

The subject of this thesis is the development of a cost calculation tool for modular power plant control panels. The tool calculates material costs for the control panels and the amount of needed material. The commissioner is ABB Power Generation Modular Power.

1.1 Thesis background

ABB's Power Generation department delivers different control panels for modular power plants. The department offers basic solutions for their main customers' power plants. It is possible to add option packages to the control panel's basic solutions, which enables the control panels to be customized for the end customer.

Basic and option packages have annually fixed prices. The fixed prices are for the complete solution packages, not for specific components. The control panel manufacturer wants to know the number of components included in the packages, before the design is finished, to be able to meet the time schedule for the project.

The final bill of material needs to be compared with the materials included in the solution packages, which has annually fixed prices. The materials in the solution packages should be included in the final bill of material.

By developing a cost calculation tool, it is possible to save money and time in the design and manufacturing stage of the project. The cost calculation tool should create a material list with prices for solution packages, based on the scope of supply and compare with the final bill of material that is created with the design tool.

1.2 Thesis objectives and delimitations

The main objective of this thesis is to create a cost calculation tool. The process to get the cost and material, for the complete control panel package should be simple and efficient. When using the cost calculation tool it should also be easy to see if materials and costs in the solution packages differ from the final bill of material.

The thesis covers materials and some material costs for the control panel. Manufacturing costs are also included in material costs. However, the thesis does not cover the cost of components for control panels, that ABB purchases. Man hours for design work are not included.

This thesis will only include the cost calculation for Local control panel for auxiliary systems.

1.3 Commissioner

The commissioner of this thesis is ABBs Power Generation department, which is part of the industrial automation division. The Power Generation department in Finland is mainly based in Strömberg Park in Vaasa and employs around 80 people. The department designs and delivers electrical-, automation-, instrument- and supervision control systems for gas, hydro, thermal and nuclear power plants. (ABB Oy, 2017c)

ABB (Asea Brown Boveri) was established in 1988 when Asea and BBC merged into one company. Today ABB operates in several fields in over 100 countries, employing around 135 000 people. The company has a revenue of 35.5 billion USD. (ABB Oy, 2017b)

In Finland ABB operates in about 20 different locations and has about 5000 employees. The revenue is 2.2 billion € and in 2016 the company invested 129 million € in R&D. (ABB Oy, 2017a)

2 Theory

This chapter looks at the theory that is the base for this thesis. The theory will include cost calculation, power plants, control systems, Microsoft Excel and Visual Basic for Application.

2.1 Modular Power Plants

All power plants are individual, no power plants are exact copies of each other. Traditionally all power plants were uniquely designed to match the requirements of the customer. By using a modularization concept when designing the power plant components, the industry is using a sell, build approach rather than sell-, design-, build- approach that was commonly used previously. (Schimmoller, 1998)

Modular power plants are built on a base model, with standard options and modifications. The base model gives the client a fully functional power plant that meets requirements. It is possible for the customer to add standard options to the base model, from a selection that the power plant manufacturer has specified. The modification part in the design is needed to handle site- and client- specific issues such as seismic conditions and various local regulations. (Schimmoller, 1998)

2.1.1 Power plant automation systems

The main function of automation systems is to monitor that various parts of the process operates at desired set points and to alarm, take trip action or shutdown in case of problems. An automation system should also monitor the most important parameters for continuous improvement. (ABB Oy , 2013) The essential parts in automation systems are hardware, software and communication links. (Shell & Hall, 2000)

2.1.2 Control system

A control system is a smaller part of the entire automation system, previously control systems were pneumatic controllers. Today they are microprocessor-based systems. (ABB Oy , 2013)

Interconnected components that form a system configuration and provide desired system response is called a control system. Components or processes that need to be controlled can be represented by blocks. Control systems can be either open-loop or closed-loop feedback control systems. (Dorf & Bishop, 2011)

An open-loop control system gives desired response by using a controller and an actuator, an open-loop system does not have feedback. (Dorf & Bishop, 2011)

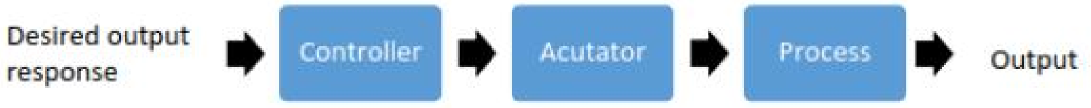


Figure 1 Open loop control system represented by blocks (Dorf & Bishop, 2011)

A closed-loop control system is a feedback system that compares the actual output to a desired output and references the input to control the process. The difference between the desired output and actual output is usually the error in the process. Feedback control systems can be single-loop feedback systems but usually they consist of more than one feedback loop. Multivariable control systems are systems where several variables must be considered in the control scheme. In the past decade these kind of systems have become more common, when system complexity and interest to achieve optimum performance has increased. (Dorf & Bishop, 2011)

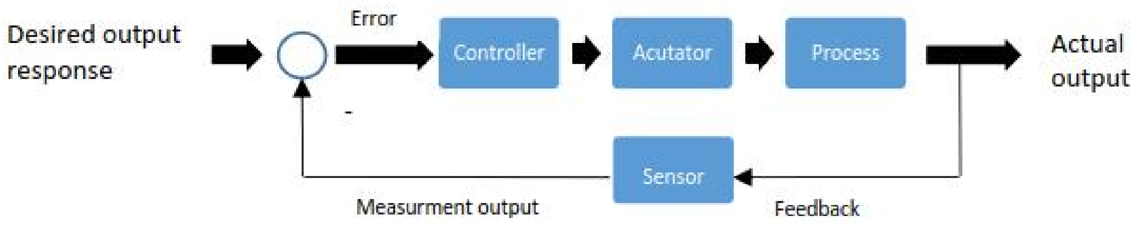
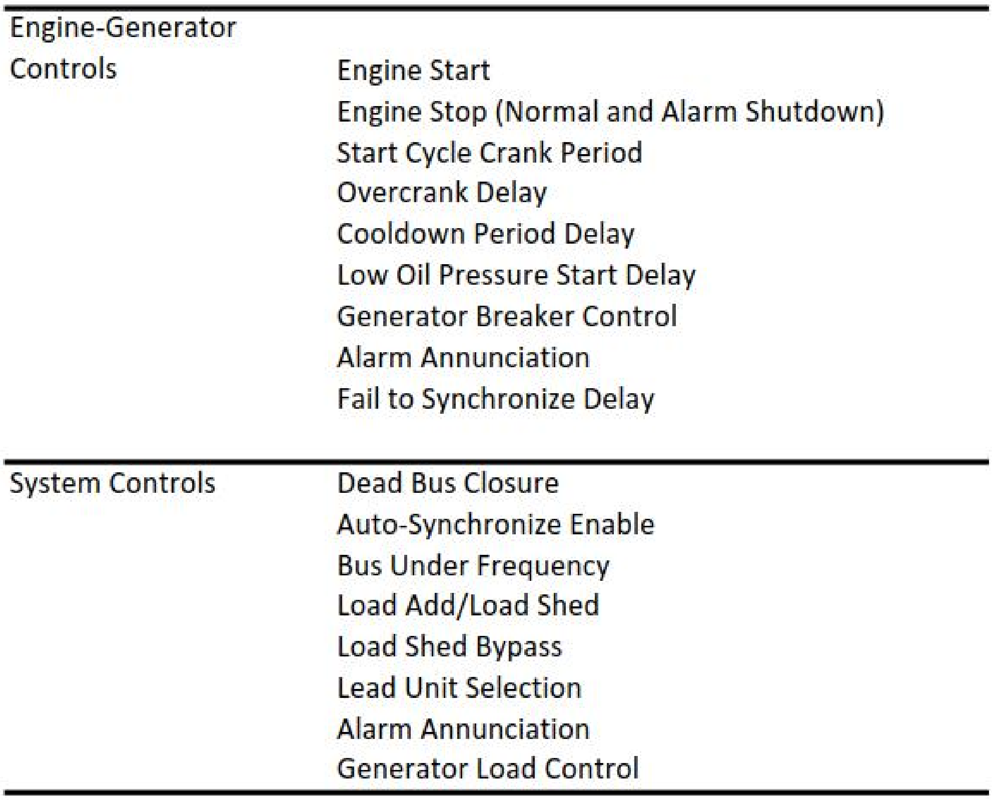


Figure 2 Closed loop control system represented by blocks (Dorf & Bishop, 2011)

On-site power control systems typically include equipment shown in Figure 3.



Engine-Generator Controls	Engine Start
	Engine Stop (Normal and Alarm Shutdown)
	Start Cycle Crank Period
	Overcrank Delay
	Cooldown Period Delay
	Low Oil Pressure Start Delay
	Generator Breaker Control
	Alarm Annunciation
	Fail to Synchronize Delay
System Controls	Dead Bus Closure
	Auto-Synchronize Enable
	Bus Under Frequency
	Load Add/Load Shed
	Load Shed Bypass
	Lead Unit Selection
	Alarm Annunciation
	Generator Load Control

Figure 3 Typical on-site power generation system controls (Leslie, 2002)

2.2 Cost calculation

A substantial portion of investment capital is invested in materials and that demands continuous and considerable attention from management. The outgrowth of excessive inventory is the reason for many business failures. However, if there are not adequate amounts of materials to meet the needs of the operating and distributing segment, efficiency suffers, costs increase, manufacturing delays become frequent, and delivery promises will not hold. (Thukaram Rao, 2000)

By scientific purchasing, the objective is to get materials at the right:

- price
- quality
- contractual terms
- time
- source
- materials
- place
- mode of transportation
- quantity
- attitude

Purchasing is a very important part of production because it directly affects on total cost. Purchasers also have to take care that the right materials, in the right amounts are received so that production is continuous. (Thukaram Rao, 2000)

There are several methods used to determine cost, and different methods are better suited to the needs of certain industries than others. The two methods of costing are *job costing* or *job order costing* and *process costing*. Other methods used are variations between these two. Different types of costing techniques can be used for special purposes to control costing and can be used irrespective of the costing method being used. (Arora, 2009) Different types of costing methods and techniques are shown in Figure 4.

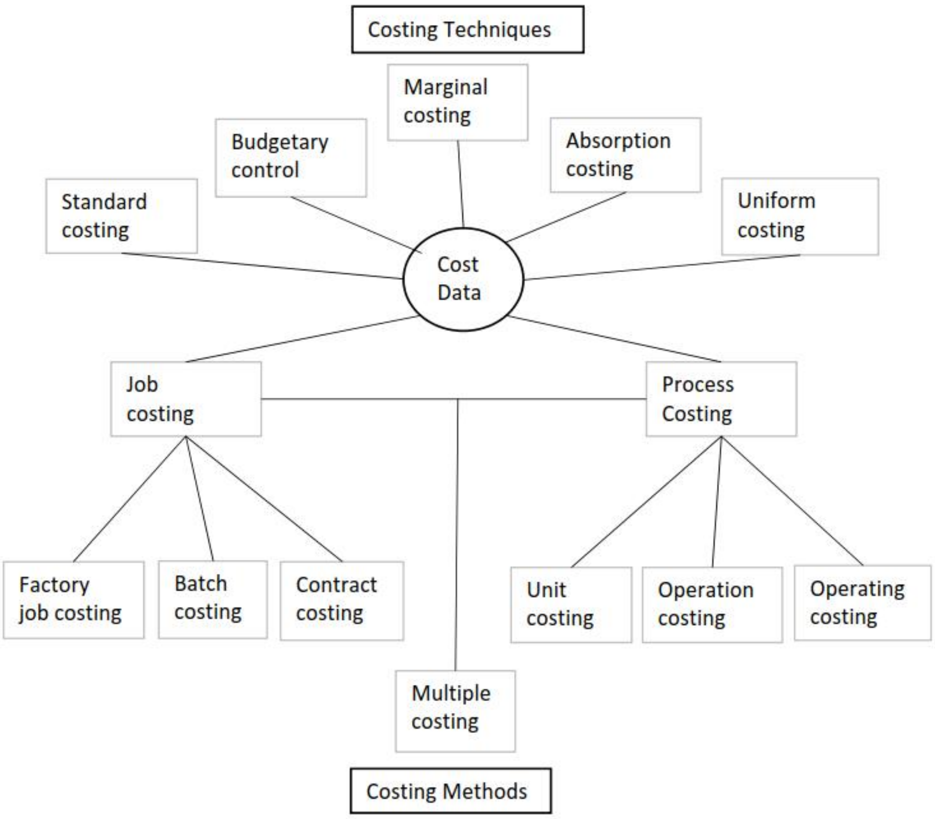


Figure 4. Costing Method and Techniques (Arora, 2009)

Costing methods and techniques are tools and should be regarded as such. None of the particular methods or techniques are superior to any other. (Arora, 2009)

2.3 Microsoft Excel

The first electronic spreadsheet program VisiCalc came in 1978 and was made for the Apple II computer. The first Excel program came in 1985 and was made for Macintosh computers. Microsoft's first spreadsheet program was Multiplan and was released in 1982 for computers running on CP/M operating system, Multiplan evolved to become Excel. Excel for windows came in November 1987 and was labeled Excel 2.0. Windows was not in widespread use when Excel 2.0 was launched so this version included a run time version of windows with just enough features to run Excel. (Walkenbach, 2010)

Macro programming in spreadsheets started as early as 1983, the Lotus 1-2-3 program had macro capability that enabled the user to record their keystrokes and that way automate many procedures. Macro language become available in Excel 2.0 in 1987, the macro language was XLM and consisted of functions that were evaluated in sequence.

The first macro language for Excel was difficult to learn and use. Visual Basic for Applications (VBA) macro programming came in Excel 5 in 1994. Excel 2010 still support XLM macros. (Walkenbach, 2010)

2.3.1 Introduction to Visual Basic for Applications (VBA)

VBA in Excel is a macro language, macro programming is used in Excel if the user wants to customize and automate Excel.

To create and use simple VBA macros the user does not need to be an advanced user of Excel. It is possible to use the macro recorder and that way record the actions and turn them into a VBA macro, when executing a recorded macro Excel does the steps the user did when the macro recorder was on.

VBA macro code can also be written in the Visual Basic Editor, creating VBA macros this way is more suited for advanced users. When creating macros in Visual Basic Editor it is possible to make macros for tasks that can not be recorded, it is also possible to create special-purpose add-ins. (Walkenbach, 2013a)

By using VBA macro the user can for example.

- Automate repetitive operations. For example, if the same work needs to be done on several worksheets a macro can do this for the user.

- Create custom commands. For example, several Excel commands can be made to execute from a keystroke or a single mouse click.

- Automate frequently performed procedures. For example prepare a month-end summary.

These are just a few examples at possibilities with macro programming. (Walkenbach, 2013a)

2.3.2 Visual Basic for Applications (VBA) programming

There are two types of VBA macro, *Sub* and *Function*, these can also be called procedures. A Sub procedure is a new command that either the user or another macro can execute. A Function procedure always returns a single value, it can be executed by other VBA procedures or used in worksheet formulas, just as built in worksheet functions in Excel. (Walkenbach, 2013a)

Recorded or written macro code will be in a module, the module can then be edited and viewed in the Visual Basic Editor. The VBA module will consist of Sub procedures or function procedures. (Walkenbach, 2013b)

Using the Excel macro recorder is very useful, it is not always the best approach and some macros can not be recorded. But by recording the steps the user want the macro to do, will almost always lead the user in the right direction. The recorder is most suited for more simple, straightforward macros. The macro recorder can not be used to create, Function procedures, it only records Sub procedures. (Walkenbach, 2013b)

When preparing to record, users should think about what the macro should do. The success of a recorded macro depends on five factors:

- How the workbook is set up while recording the macro
- What is selected when starting to record
- The use of either absolute or relative recording
- How accurate the recorded actions are
- In which context the macro is to be played back

Excel normally records absolute references to cells when the user starts recording. Excel records in absolute recording mode as a standard. When recording in this mode Excel uses absolute references. This means that if a recorded value is in cell B1, when running the macro Excel will always put that value in B1. (Walkenbach, 2013b)

It is also possible to record in relative mode. When recording in relative mode Excel works with the cell locations in a relative manner. To use relative mode the user needs to change recording method in the developer tab.

Relative recording means that when running the macro it will always start from the active cell. Recording method can be changed at any time even in the middle of recording. (Walkenbach, 2013b)

When recording, Excel can store macros in different places, by default macros get stored in a module in the active workbook. When choosing to store macro in a new workbook Excel opens a blank workbook. If the recorded macro will be used in multiple workbooks, it is good to store the macro in your personal Macro Workbook, it is a hidden workbook that opens automatically when Excel starts, and this workbook does not exist before it is specified as location for a recorded macro. (Walkenbach, 2013b)

Recorded macros are not the most efficient, often the recorded macro creates a lot of extra rows in the code which slows it down. Despite this the macro recorder is very useful, the recorder is an excellent tool in starting to understand and create macros. Recorded macros can be modified to so they work very well, but that requires some experience in VBA macros. (Walkenbach, 2013b)

For the following tasks the macro recorder can not create code.

- Performing any type of looping
- Performing any type of conditional actions
- Assigning values to variables
- Specifying data types
- Displaying pop-up messages
- Displaying Custom dialog boxes

VBA uses many elements common to all programming languages, it is a real live programming language. VBA uses elements like: comments, variables, constants, data types, arrays and several others. (Walkenbach, 2013b)

Comment

The simplest VBA statement is a comment. These statements are ignored by VBA so they can consist of anything. Usually comments are used to describe what the code does.

Comments begin with an apostrophe ('), any text that comes after an apostrophe in a line of code is ignored by VBA. There is one exception to this and that is an apostrophe inside a set of quotation marks, VBA does not interpret this as a comment. (Walkenbach, 2013b)

Effective use of comment is to identify the author, describe the purpose of each sub or function procedure briefly, keep track of changes made to the procedure and describe used variables if they are not used in a understandable way. (Walkenbach, 2013b)

Variables

When the code is running it needs to store temporary data somewhere, variables are used for this purpose. When declaring variables use the Dim statement before the variable as example Dim variable name [As type].

Variable names must follow these rules:

- They must begin with a letter
- They must contain only letters, numbers, or the underscore character (no spaces)
- They must not exceed 40 characters (Excel 2007)
- They must not be a reserved word

Variables do not have to be declared before using them but it can lead to errors in the code if the variable later in the code is misspelled, because VBA does not know which variable it should use.

By using explicit declaration, misspelling issues in the code do not exist because VBA will give an error message whenever it encounters an undeclared variable. This is achieved by typing in Option Explicit in the declarations section, this works on per-module basis.

There are different types of variables local variable, module-level variable and global variable.

Local variables are declared by using Dim, Static or ReDim (arrays only), these types of variables declared as Dim stay in existence only as long as the procedure is executing. Variables declared as Static stay in existence for the lifetime of the application.

Module-level variables are declared by using Dim, Static or ReDim(arrays only). These types of variables are available to all procedures within that module but not to the rest of the application. They remain in existence for the life time of the application and preserve their value.

Global variables are declared with the Global statement in the declaration part of the module. These can be accessed by any code in the application they exist and retain their value for the lifetime of the application. (Sheperd, 2010)

Constants

When in need of hardcoded values or strings it is very good programming practice to use constants. The value of variables varies but for constantans, they do not. Constants are useable if a specific value is needed several times then it is possible to refer to the constants name instead of a specific value. (Walkenbach, 2013b)

Data types

There are several different data types, variables can be given a data type, that determines which kind of data it can store. By using the right or wrong data type in the code, it is possible to have impact on the efficiency of the code. If variables are declared without a specified data type it will by default be Variant. (Sheperd, 2010)

Variant can store all kinds of data numbers, text, dates or other information. Variant variables can also freely change type during runtime. Performing mathematical operations on variants that include other than numerical values gives a Type mismatch error. (Sheperd, 2010)

When working with numeric types, that are whole numbers the variable should be declared as Integer or Long. When working with fractions of numbers the variable should be declared as Single, Double or Currency.

String type variables always contain text, string type variables can be manipulated in different ways. (Sheperd, 2010)

Arrays

Groups of variables that share a common name are an Array. Most programming languages support Arrays. Unlike variables, Arrays always need to be declared. Arrays can be one- or multi-dimensional. (Walkenbach, 2013b)

One-dimensional arrays can be compared to a column in a spreadsheet and multidimensional arrays can be compared to a spreadsheet with columns and rows, this is called a two-dimensional array. A multidimensional array can also be three-dimensional, and then it can be compared to columns, rows and spreadsheets. (Sheperd, 2010)

Arrays can be declared with a specific number of elements or they can be declared as dynamic that changes number of elements according to needed amount of elements. (Sheperd, 2010)

Conditional statements

Decision making or conditional statements is one of the most important areas in programming because it specifies what will happen when different events occur. A computer program that never do any decisions would be very dull. (Sheperd, 2010)

Looping

Programs would be very slow and difficult to maintain without looping. By using looping a block of code is allowed to repeat until a condition or a specific value is met, without need to write out the code several times with different values. There are several different types of loops which have different kinds of structure. (Sheperd, 2010)

Operators

Mathematical functions, comparison functions and logical functions are performed by operators.

Mathematical functions can be plus/minus, multiplication/division and so on. Comparison functions can be less than, greater than, equality and so on. Logical functions use pure binary math to decide the result, they do this by a logical bit-by-bit conjunction on two expressions. Logical functions are and, not, xor and so on. (Sheperd, 2010)

3 Project realization

In this chapter, the realization of this thesis work will be described. The project realization part includes a Program specification and Program realization.

3.1 Program specification

A program specification is a description of the different functions that is demanded from the thesis work. The specifications on the program was set in an early stage of the thesis but the specifications have evolved throughout the whole realization process of the thesis.

3.1.1 Program description

The cost calculation tool calculates the material cost for the control panel, lists standard materials and specific electrical consumer groups for the control panel. The program sorts the material on a specific material sheet. The program compares the material list to the bill of material that is generated in the drawing program and highlights the differences, so they are easy to spot. See Figure 5 for a mind map on the working order of the program. The most important calculations in the program are price and amount of material.

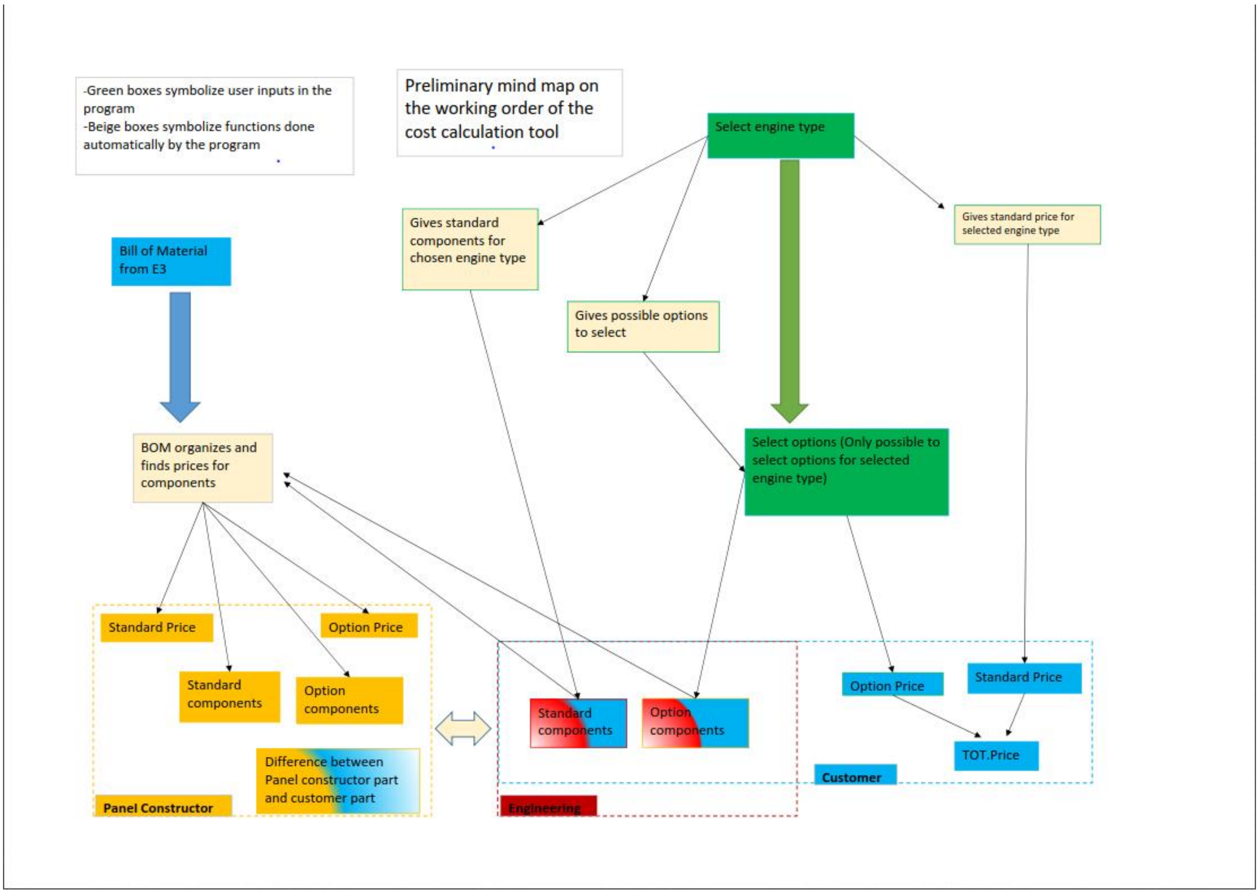

-Green boxes symbolize user inputs in the program
-Beige boxes symbolize functions done automatically by the program
Preliminary mind map on the working order of the cost calculation tool
Select engine type
Gives standard components for chosen engine type
Gives standard price for selected engine type
Bill of Material from E3
Gives possible options to select
BOM organizes and finds prices for components
Select options (Only possible to select options for selected engine type)
Standard Price
Option Price
Standard components
Option components
Difference between Panel constructor part and customer part
Panel Constructor
Standard components
Option components
Engineering
Option Price
Standard Price
TOT.Price
Customer

Figure 5 Preliminary mind map on the working order of the cost calculation tool

3.1.2 Tools to be used

The cost calculation tool is created using Microsoft Excel. To get a fast and automatized calculation tool macro programming in Visual Basics for Application will also be used.

The user interface of the tool will be done in English.

3.1.3 Worksheets in the cost calculation tool

Worksheets that needs to be included in the cost calculation tool:

-**Start sheet.** The start sheet is the page where the user does his input in the program. This page needs to be well organized and easy to use. Engine type, options for the specific engine type and PLC equipment for the control panel are selected on this page. The input on this sheet will decide what components will be included in the control panel. There will be needed a button that shows options for specific engine type, and a button that creates the content to the price sheet.

-**Price sheet.** The program needs to include a sheet that lists all materials that are needed to construct a control panel according to the criteria's selected on the start sheet. The materials needs to be listed in groups such as; standard material, specified electrical consumer groups and PLC material. The listed groups need to include prices specified per group so that it corresponds with the annually agreed price list, that is based on prices for the different groups and not component specified prices. A total price for the control panel is also needed.

A button that creates a material sheet based on materials in this sheet is also needed.

-**Material sheet.** A sheet that organizes and lists materials from the price sheet so that it can be easily used to see what materials are needed for the manufacturing of the control panel. This sheet will not include any prices only materials and quantities.

A button that compares the material sheet against the bill of material sheet, is to be placed on this sheet.

-**Bill of material sheet**. This is an optional sheet that is created in the drawing program, the user adds this sheet to the program if comparison to material list is needed.

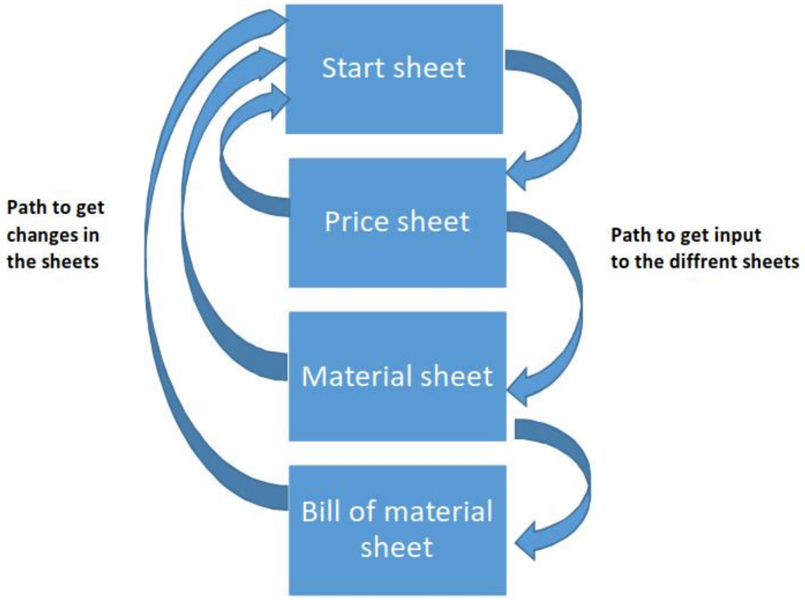


Figure 6. Illustration on how input affects different sheets

3.2 Program realization

Program realization is an answer to the program specification. This chapter will go through how the project is executed to meet the demands in the program specification.

The program is realized in Microsoft Excel by using macro programming in Visual Basic for Applications.

3.2.1 Overall solution

The overall solution is done by dividing the whole program in to smaller pieces and creating every piece as a separate program. When one piece works I start on next piece and modify the working pieces so they work in harmony with the new piece.

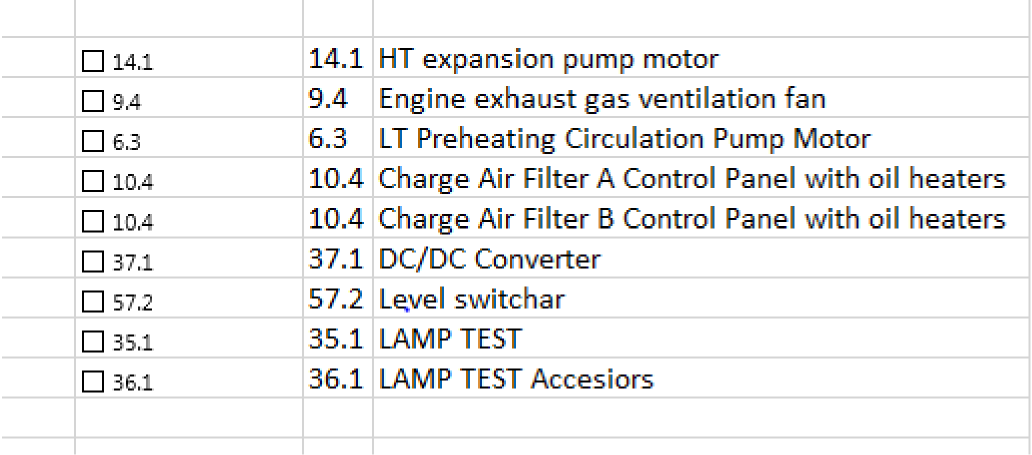
The first step was to get a clear idea of how the program should work between the different worksheets and from that, start to get an idea on how to specify what material is needed for the control panels for different engine types.

It was decided in an early stage that the annually fixed price list would be used as the base sheet, from where the program takes the materials that are inserted on the price sheet, because it is already sorted in groups with material and price listed per group.

The base sheet will include every possible standard- and option group. This means that the standard and option groups, for the specific engine type selected, needs to be specified somehow. To solve the issue it was decided that the customers electrical consumer list with small modifications would be used to sort out what standard groups and options are specific for each engine type. For each engine type the customers specific electrical consumer list with modifications, was added as separate sheet.

The base sheet and the electrical consumer sheets are data storage sheets so the user will not need to modify them in normal use of the program. They will still be left unlocked so it is possible to do modifications in them.

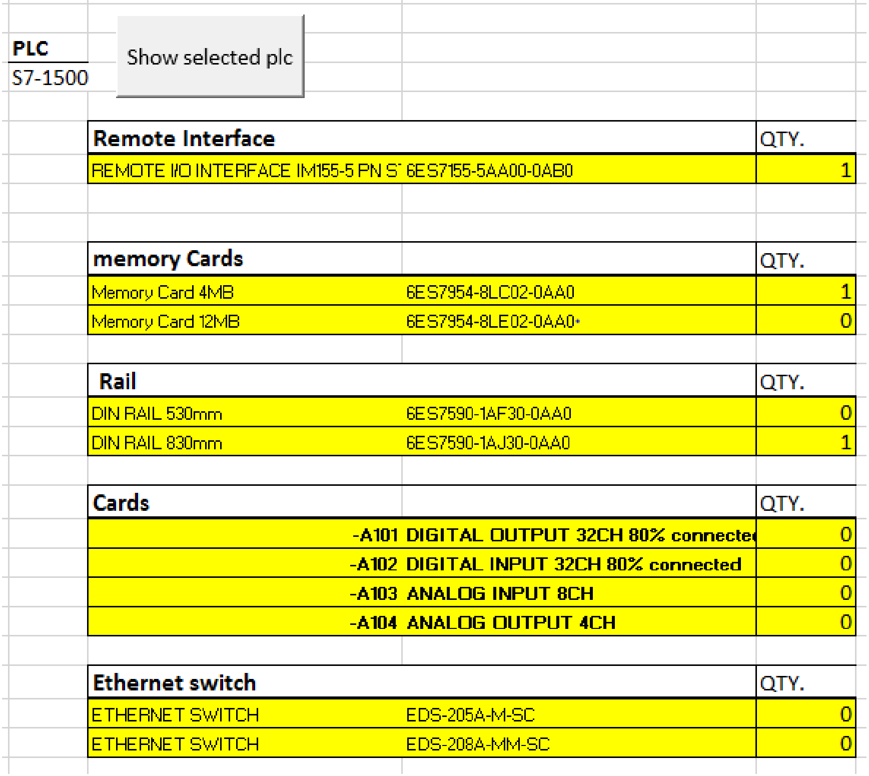
The start sheet is the sheet where the user does his input to the program and it will affect the output directly. The most important input on the start sheet is the engine type, it is selected from a dropdown menu. A button that shows options when clicked is placed on the start sheet, by clicking on this the available options for the selected engine type will be shown with checkboxes to select options that should be included in the control panel.



☐ 14.1	14.1	HT expansion pump motor
☐ 9.4	9.4	Engine exhaust gas ventilation fan
☐ 6.3	6.3	LT Preheating Circulation Pump Motor
☐ 10.4	10.4	Charge Air Filter A Control Panel with oil heaters
☐ 10.4	10.4	Charge Air Filter B Control Panel with oil heaters
☐ 37.1	37.1	DC/DC Converter
☐ 57.2	57.2	Level switchar
☐ 35.1	35.1	LAMP TEST
☐ 36.1	36.1	LAMP TEST Accesiors

Figure 7 Example on options available to select

Different types of PLC's are used and these are not engine specific. On the start sheet there is a drop down menu to choose PLC and a button to change to the PLC selected in the dropdown menu. For the selected PLC, all the available material for the specified PLC is shown. To select PLC material the user fills in the quantity of the specific material in the quantity field. The PLC was created this way because the amount of PLC material varies per project and per engine type.



PLC		
S7-1500		
Remote Interface		QTY.
REMOTE I/O INTERFACE IM155-5 PN S' 6ES7155-5AA00-0AB0		1
memory Cards		QTY.
Memory Card 4MB	6ES7954-8LC02-0AA0	1
Memory Card 12MB	6ES7954-8LE02-0AA0	0
Rail		QTY.
DIN RAIL 530mm	6ES7590-1AF30-0AA0	0
DIN RAIL 830mm	6ES7590-1AJ30-0AA0	1
Cards		QTY.
	-A101 DIGITAL OUTPUT 32CH 80% connecte	0
	-A102 DIGITAL INPUT 32CH 80% connected	0
	-A103 ANALOG INPUT 8CH	0
	-A104 ANALOG OUTPUT 4CH	0
Ethernet switch		QTY.
ETHERNET SWITCH	EDS-205A-M-SC	0
ETHERNET SWITCH	EDS-208A-MM-SC	0

Figure 8 PLC material selection on the start sheet

There is a button on the start sheet to insert the needed material to the price sheet. Clear options button and clear price sheet button are also on the start sheet so the user can clear the program so it is possible to start from zero.

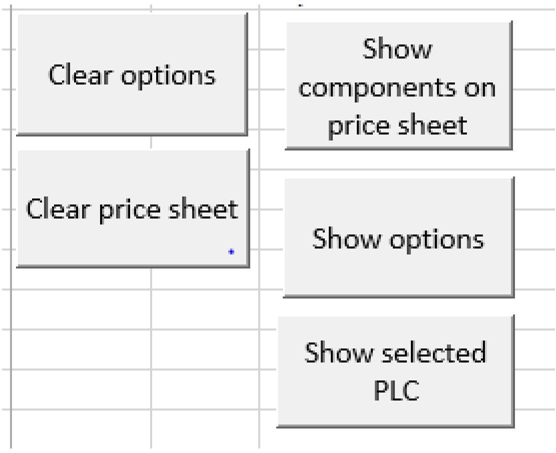


Figure 9 Buttons on the start sheet

A price sheet is available in the program, but it is empty if the user has not selected any input on the start sheet and inserted the material to it. The material is sorted on the price sheet so standard material comes first, then options and lastly PLC material. Prices are shown per standard and option group in a column after the group. The total price for the panel is shown also on this sheet. On this sheet there is a Clear price button, that also empties the price sheet. The material sheet is also created from this sheet with the button create material sheet.

The material sheet is a sheet that only includes material, quantity and supplier. This sheet is created based on what material there is on the price sheet. The material is sorted by supplier first and then by material so its easy for the panel manufacturer to see what materials they should deliver. A button to compare the material list with the bill of material is placed on this sheet. To compare the material list to the bill of material, the user needs to add the project specific bill of material sheet manually, after the material list.

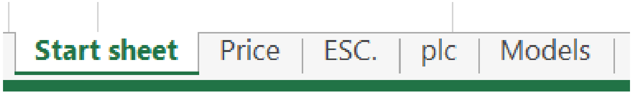
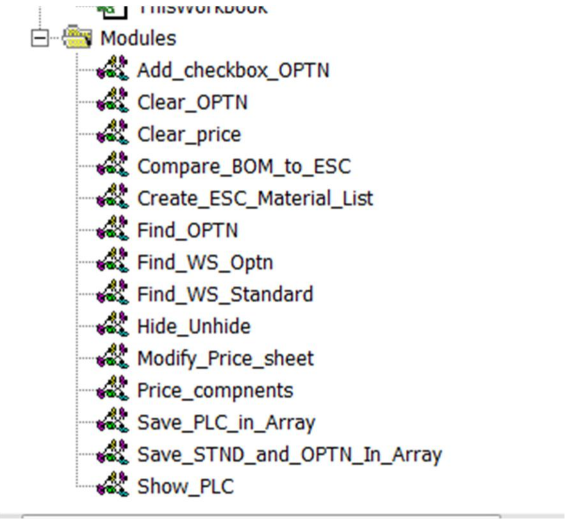


Figure 10 Sheets that the user uses and can modify

3.2.2 Program code structure

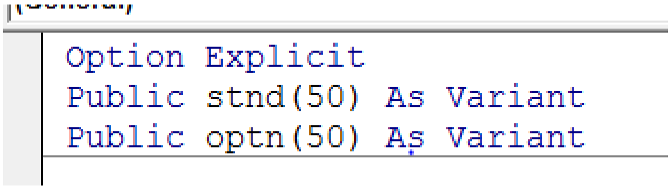
When it was decided that the program would be created as several small programs that work together as one program, it was natural to make the code structure as small modules that are more or less own programs. Since the code structure was created this way, the amount of modules is 14pcs.



- Modules
  - Add_checkbox_OPTN
  - Clear_OPTN
  - Clear_price
  - Compare_BOM_to_ESC
  - Create_ESC_Material_List
  - Find_OPTN
  - Find_WS_Optn
  - Find_WS_Standard
  - Hide_Unhide
  - Modify_Price_sheet
  - Price_compnents
  - Save_PLC_in_Array
  - Save_STND_and_OPTN_In_Array
  - Show_PLC

Figure 11 Modules in the program

By structuring the code as smaller modules, it is easier to start working on the code when just a small code piece is to be considered at time. An issue with making the modules as small programs that need to interact in harmony as a program, is that it can demand quite a lot of changes in a working module. A way to fix the modules so they interact together is to declare the procedures that interact between the different modules as Public.



```
Option Explicit
Public stnd(50) As Variant
Public optn(50) As Variant
```

Figure 12 Public declarations of arrays

The modules are created so that they call for the next specified module when specific criterias are reached in the module that is running, this makes the program work automatically between the modules that create the wished output from the selected input.

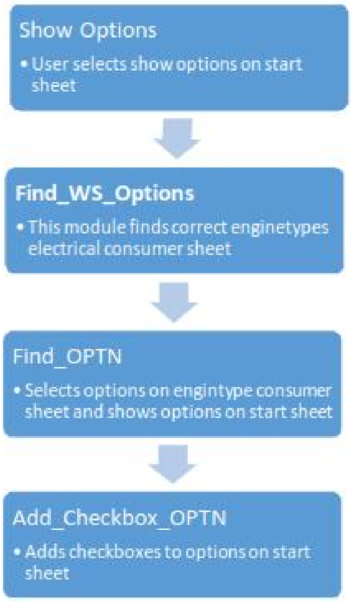

Show Options
• User selects show options on start sheet
Find_WS_Options
• This module finds correct enginetypes electrical consumer sheet
Find_OPTN
• Selects options on engintype consumer sheet and shows options on start sheet
Add_Checkbox_OPTN
• Adds checkboxes to options on start sheet

Figure 13 Example on how modules work together to show options

3.2.3 Creation of the program code

When creating the program code, the structure of the code has had quite a big impact on how the code should be created so it interacts in harmony between the different modules. Before starting to write the code the primary thing is to know what the code should do, so the first thing was to decide what that specific module should do.

Several of the methods mentioned in the Theory Chapter have been used when creating the code. Most of the code has been created by mixing these different methods and, by that way creating a working solution.

The recording of macros has been used quite a lot on more simple tasks. To use the macro recorder and to modify the code to work as wanted is very effective. The recorder has also been used a lot to create pieces of the more complex code. When using the recorder it has been important to specify precisely what it should do, otherwise a lot of unnecessary code would be included, that would need to be cleaned away.

To write macro code from scratch demands that you are quite familiar with coding in Visual Basic for Applications even if it is not a difficult coding language. Luckily there is a lot of people in the world who work with coding, so there is a lot of help to find when needed. When creating the written code for this program I have founded much help and examples on how to create macro code on the internet. Unfortunately the code examples found on the internet are not specifically made for this program so they always need modification.

The programs' data search works mostly in a specific column, so to start searching through a database with approximately 2000 rows would be quite slow if manually comparing one value at time between sheets or saving the selected values in variables.

For data saving in the program has arrays been used. Based on the input from the start sheet the program searches for, data to save in to arrays by using different *Do loops* and *if* statements. The data is saved in arrays to get the program to work fast and efficiently.

Depending on the inputs the arrays work differently, all used arrays are one-dimensional, so it is easy to loop through the arrays when searching in the base sheet groups, what to be inserted to the price sheet. Some of the data saving in the arrays work, as there are several arrays parallel so that a components' different info is in four different one-dimensional arrays, but at the same position in every array.

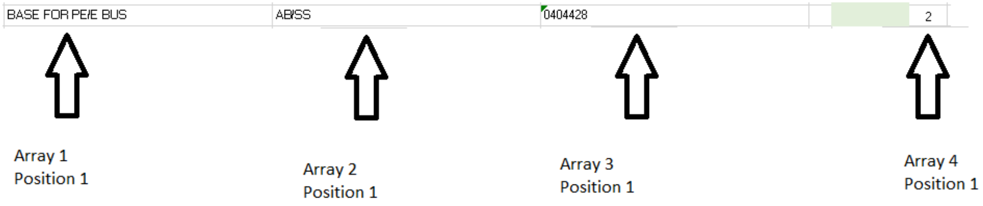


Figure 14 Illustration on how inputs in different arrays work parallel

The creation of the program code has been easy in some modules while other modules have been very difficult and time consuming.

The code for comparing the material list with the bill of material is not ready at the moment so it is hard to say if it will result in a more complicated code than already is used in the program.

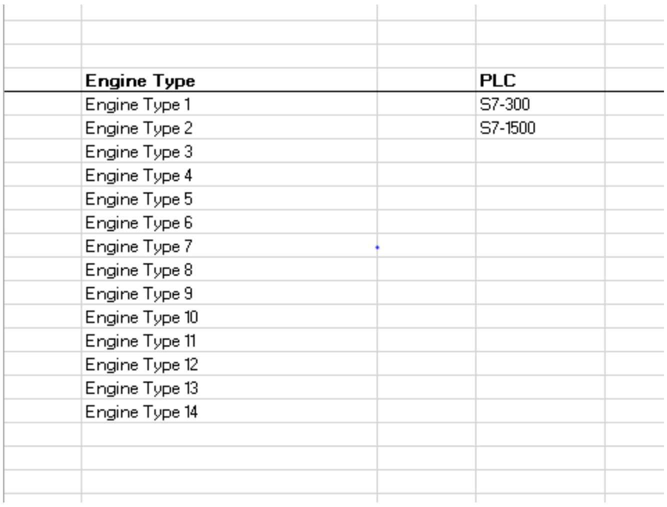
3.2.4 Changes in the program

The program will of course evolve and change when its used, engine types will be added and removed, the base sheet with prices will be revised each year and of course other changes will happen to the program also.

The base sheet will most probably be changed every year so it, has the latest price and components. The program is created so that the base sheet can be changed or revised. The most important thing, is that the same info is found in the same column because the program searches for the specified standard and option groups in a specific column. Rows can be added and removed as the user wishes, it will not affect the program.

PLC material is stored on an own sheet called PLC because they are not specified to some engine type. On the PLC sheet it is possible to add and remove PLC materials in the different PLC's that are specified in the program as long as the added material has the same structure as the existing ones. If more PLC types are added the code needs to be changed to include that model, but changing an existing PLC to another and inserting the new one with same structure as the old one is possible. When adding, removing or changing PLC type the user needs to do changes on the models sheet, in Figure 15 the models sheet is shown.

When adding engine types the user needs to add the engine type on the model sheet (adds the engine type in the dropdown menu on the start sheet) and insert the electrical consumer list for the added engine type. The engine wise consumer lists needs to have the same structure as existing ones. When modifying existing engine wise consumer lists the data needs to be in same columns as before.



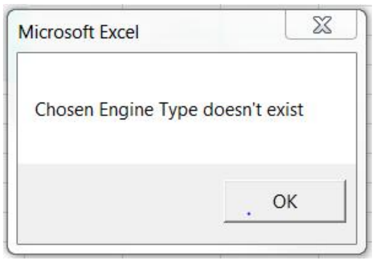
Engine Type	PLC
Engine Type 1	S7-300
Engine Type 2	S7-1500
Engine Type 3	
Engine Type 4	
Engine Type 5	
Engine Type 6	
Engine Type 7	
Engine Type 8	
Engine Type 9	
Engine Type 10	
Engine Type 11	
Engine Type 12	
Engine Type 13	
Engine Type 14	

Figure 15 Models sheet where user can do changes

3.2.5 Further improvements

The program is not 100% finished when writing this thesis it is about 90% done, one of the most important further tasks is to get the material list to be compared with the bill of material.

Another big task is to add message boxes that informs the user if something is wrong. Some short info messages could be added so that the user could click on a button and a message box pops up, that shortly explains what to do on specific sheet.



Chosen Engine Type doesn't exist

Figure 16 Illustration on how message boxes could look

A modification on the code that shows PLC material, to make it possible to add more PLC types without need to change code.

The program could also include the possibility to do these same calculations on other control panels for power plants.

Some of these mentioned improvements will be implemented in the program as the program is getting finished. A manual for the program is also a improvement that probably will be made.

4 Result

The expected result for this thesis work was an efficient control panel cost calculation tool that creates a price list with specified prices for standard packages, option packages and total price for the control panel. The expected result included also a material list and possibility to compare material list to the finished bill of material. The end result is a automatized control panel cost calculation tool created with macro programming in Microsoft excel.

The cost calculation tool does not fulfill all demands in the program specification, it fulfills the following requirements:

- There is one start sheet were all user inputs are made.
- Based on the user inputs, the program automatically inserts the materials in groups on the price sheet with price per group and total price.
- A material list can be created automatically from the price sheet.

These following requirements are not fulfilled by the program:

- Comparing the material list to the bill of material.

The requirement that are not fulfilled will be included into the program before it will be taken in to active use.

Testing has been done throughout the whole creation process of the project. When a module or procedure in a module has been done it has been tested. Every new module that builds on an existing module has been tested together with others as a whole procedure, to find out errors. When testing the program like this, in smaller pieces it has been easier to find the errors in the code. Errors that have been very time consuming have also occurred when testing, these errors have been of such kind that the error occurs in the module that is under testing but the reason for the error is in a different module.

I have so far done all testing of the program. Now when the program is this close to completion the next step, would be to give a pilot version to a couple of designers to see what improvements they would want on the working functions in the program.

The cost calculation tool gives the user an easy and fast way to see all materials and cost of the control panel. I hope that the cost calculation tool will be useful for the commissioner and will be used actively for achieving their goals.

5 Discussion

When I thought about writing a thesis work two years earlier I never imagined that I would do a thesis that is 90% coding. Now I have done it and have to say that it has been educational for me.

To create a program from scratch is not easy at all but thanks to the help I have got from my supervisor at ABB and the help on the internet it has been a somewhat easier.

The user interface of the program is still a untested area when it is only me who has used the program. I am satisfied with the user interface at the moment, especially that all inputs are made on the start sheet. Now when I give a pilot version of the program to a couple of designers I will surely get some feedback.

The program is made so it is quite easy to do the modifications that can be predicted. However, when an unpredicted modification is needed, it means that the code needs to be changed and to do that, it has to be someone who has some expertise in macro programming in Microsoft Excel. The code is structured so that it should be quite easy to find the procedure that needs the change but of course, it seems easy for me who has created the code.

In the writing moment the program is not 100% ready the last part of it is under work, the complexity of the program made it very time consuming, but it will be finished shortly.

The boundaries for this program have been made narrower throughout the creation process of the program, when I realized that it was not as easy as I had thought in the beginning. When I started creating the program a lot of the time went into learning how to program macros in Microsoft excel, it was a completely new area for me. I highly value the knowledge that I have gathered in programming while doing this thesis work.

Bibliography

ABB Oy , 2013. *Power Generation.* [Online]
Available at: http://new.abb.com/power-generation
[Accessed 8 January 2017].

ABB Oy, 2017a. *ABB Suomessa.* [Online]
Available at: http://new.abb.com/fi/abb-lyhyesti/suomessa
[Accessed 3 January 2017].

ABB Oy, 2017b. *ABB Who we are - ABB in brief.* [Online]
Available at: http://new.abb.com/about/abb-in-brief
[Accessed 03 January 2017].

ABB Oy, 2017c. *Power Generation.* [Online]
Available at: http://new.abb.com/fi/abb-lyhyesti/suomessa/yksikot/power-generation
[Accessed 3 January 2017].

Arora, M., 2009. *Methods and Techniques of Cost Accounting : Theory, Problem and Solutions.* s.l.:Himalaya Publishing House.

Bolton, W., 2002. Control Systems. In: *Control Systems.* s.l.:Newnes.

Dorf, R. C. & Bishop, R. H., 2011. *Modern Control Systems.* 12 ed. New Jersey: Pearson.

Leslie, D., 2002. *On-Site Power Generation.* 4 ed. Florida: Electrical Generating Systems Association.

Schimmoller, B. K., 1998. *POWER ENGINEERING.* [Online]
Available at: http://www.power-eng.com/articles/print/volume-102/issue-1/features/power-plants-go-modular.html
[Accessed 8 January 2017].

Shell, R. & Hall, E. L., 2000. New York: CRC Press.

Sheperd, R., 2010. *Excel 2007 VBA Macro Programming.* New York: McGrawHill.

Thukaram Rao, M., 2000. *Elements of Costing.* s.l.:New Age International Ltd., Publishers.

Walkenbach, J., 2010. *Excel 2010 Power Programming With VBA.* Indianapolis: Wiley .

Walkenbach, J., 2013a. *Excel 2013 Bible.* Indianapolis: John Wiley & Sons.

Walkenbach, J., 2013b. *Excel VBA Programming For Dummies.* 3 ed. Hoboken: John Wiley & Sons Inc..