Massey University

Declaration Confirming Content of Digital Version of Thesis

I confirm that the content of the digital version of this thesis

Title: Automatisation in programming of a PLC code

is the final amended version following the examination process and is identical to this hard bound paper copy.

Student's Name: Nikola Mastilovich

Student's Signature: Mastilović N.

Date: 16/7/2010

Automatisation in programming of a PLC Code

A thesis presented in partial fulfilment of the
requirements of the degree of

Masters of Engineering
In
Mechatronics

At
Massey University
Auckland, New Zealand

Nikola Mastilovich

2010

Acknowledgments

The author would like to thank Realcold Milmech for making this research thesis possible. In addition, it is important to acknowledge the time and tools that Realcold Milmech has provided for this thesis.

A big thank you goes to Dr. Johan Potgieter for all of his help, guidance, and support in the completion of this thesis.

Finally, I owe a special thanks to my wife, Adela Vidicki-Mastilovich, for supporting and helping me achieve this goal. Your help is greatly appreciated.

Table of Contents

List of Figures

Abstract

A competitive edge is one of the requirements of a successful business. Tools, which increase an engineer's productivity and minimize cost, can be considered as a competitive edge.

The objective of this thesis was to design, create, and implement Automatic PLC Code Generator (APCG) software. A secondary objective was to demonstrate that the use of the APCG software will lead to improved project efficiency and enhanced profit margin.

To create the APCG software, the MS Excel and Visual Basic for Applications (VBA) programs were used as the platform. MS Excel sheets were used as a user interface, while VBA creates the PLC code from the information entered by the engineer. The PLC code, created by the APCG software, follows the PLC structure of the Realcold Milmech Pty. Ltd, as well as the research "Automatic generation of PLC code beyond the nominal sequence" written by Guttel et al [1].

The APCG software was used to design and create a PLC code for one of the projects undertaken by Realcold Milmech Pty. Ltd. By using APCG software, time to design, create, and test the PLC code was improved when compared to the budgeted time. In addition, the project's profit margin was increased.

Based on the results of this thesis it is expected that the APCG software will be useful for programmers that tend to handle a variety of projects on a regular basis, where programming in a modular way is not appropriate.

1 Introduction

1.1 Overview of the problem

In today's global economy, running a successful business is very challenging and competitive[2]. A competitive edge is one of the necessities to have a successful business. A product needs to be of *"good value for client's money"*, which means good quality but low cost[3]. This, also, holds true for the Industrial sector.

Because of the economic instability, Manufacturing and Infrastructure industries are reviewing their strategies to cut operational costs, which will make them more competitive and sustainable. One form of cutting operational costs is to employ automation solutions. [4]

The fact, that Manufacturing and Infrastructure industries are cutting operational costs through automation is an opportunity, as well as an indicator, that Automation companies need to become more competitive. In this essence, an Automation company needs to be more efficient, have a better quality and lower cost of a product, than the opposition.

The cost of any product that an Automation Company provides can be broken down into three parts. The breakdown of the projects cost holds true for any Engineering discipline, i.e. Electrical, Mechanical, Refrigeration etc. From the experiences of Realcold Milmech Pty. Ltd., the cost of a project, is assumed to be evenly spread between the following three parts:

1. Hardware Costs
2. Installation Costs
3. Engineering Design Costs

(NOTE: This thesis is done in conjunction with the Realcold Milmech., and is largely based on the experiences of the company.)

Hardware cost can be considered as a set cost. This cost does not greatly differ between suppliers. The client holds the right to select their preferred hardware supplier, i.e. Rockwell, Danfoss, etc.

Installation costs tend to vary between companies. Consequently, the project engineer needs to research the market to find the best price.

An area where an Automation Company can directly influence cost, and in turn incur cost savings is the Engineering Design. Greater efficiency and productivity of engineers in this area would represent a competitive edge for a company[2].

To reduce the Engineering costs, companies have developed various methods. One of those methods is to re-use the existing solutions. While this method reduces the Engineering costs, significant engineering design and time is still required.[1]

1.2 Proposed Solution

The aim of this thesis was to improve the efficiency and quality, as well as, reduce the costs of an Electrical project and thereby enhance the profit margin. There are various ways of improving costs of an Electrical Project. Therefore, specifically, the aim of the thesis is to minimize and improve the PLC code Design costs. This will be attempted by creating a software mediator that will automate programming, and improve the quality and efficiency of a PLC code.

The created software mediator, Automatic PLC Code Generator (APCG), will automatically generate a PLC code from the information entered by an engineer.

This thesis will be of interest to all Electrical Engineers. As this thesis is based on the standard of Realcold Milmech Pty. Ltd., the results may not be agreeable to other companies' standards. Having that in mind, the program may need to be adjusted to suit each company that chooses to use it.

1.3 The objectives of the thesis

i. To carry out a literature review in order to identify existing research on Automatic PLC Code Generation
ii. To define a PLC code structure which the APCG software will follow. Each automation company has its own PLC code structure. This thesis will follow the PLC code structure, which is used by Realcold Milmech Pty. Ltd.
iii. To create the APCG software which will automatically create a PLC code. The aim is to have a PLC code, which has all necessary parts to be fully functional. The APCG software will use the inputs, outputs, and sequences, which are entered by the user.
iv. Once the APCG software is created, the attention of the thesis is to test it in real life applications. The APCG software will be used and tested in an electrical project, conducted by the author, for Realcold Milmech Pty. Ltd.
v. To test and report on the effectiveness and efficiency of the APCG software. Time will be used as the outcome unit of measurement.
vi. Past projects, that are similar to the project used for testing, will be used when measuring the effectiveness and efficiency of the APCG software.

1.4 Subsequent Chapters

Chapter 2: Literature Review

Overview of the past research and achievements will be described in this chapter, which will also outline the scope of this thesis. The research question and hypotheses will be stated in this chapter as well.

Chapter 3: Method

This chapter will describe the methodology of creating the APCG software. The PLC structure, the tools used to create the APCG software and the steps taken to test the efficiency of the APCG software will be explained in detail.

Chapter 4: Functionality of the APCG software

This chapter will explain the functionality of the APCG software.

Chapter 5: The APCG software used in Real Life Project

In this chapter, effectiveness of the APCG software will be measured by recording the time it takes to create a PLC code by using the APCG software, and compare the new data with the existing data of past projects.

Chapter 6: Discussion

The results and contribution of this thesis will be explained. In addition, Limitations and Future Research will be outlined in this chapter.

2 Theory

2.1 History of the Programmable Logic Controller (PLC)

Prior to the existence of the PLCs, the control of machines was done through hard-wired systems. These systems consisted of relays, cam timers, drum sequencers, and dedicated closed loop controllers in the control panels. Unfortunately, for big projects, the amount of relays counted in thousands, thus the control panels were quite large. When a change in the system was required, the task was expensive and time consuming. Figure 1 shows the extensive size of the relay based control panel.[5]

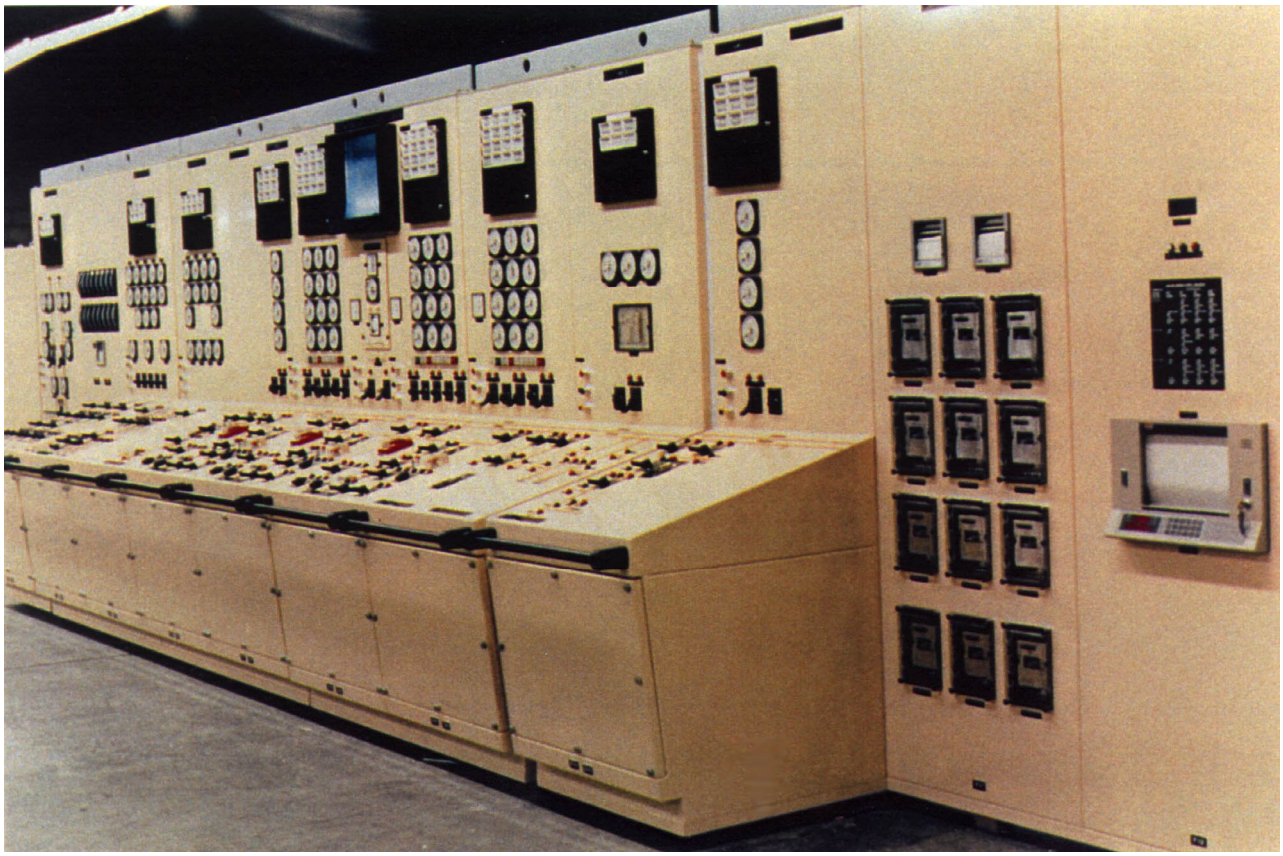


Figure 1: Relay Based Control Panel[6]

American automotive manufacturing industry was the first to realize the need for replacing relay-based controls. In 1968, Bedford Associates, of Bedford Massachusetts, created a first PLC (named 084), for GM Hydramatic (division of General Motors). The PLC was designed to replace hard-wired controls. Because, PLCs, were created for the needs of the automotive industry, it needed to withstand harsh conditions, such as dust, moisture, heat and cold.[5]

Modicon (MOdular DIgital CONtroller), company created by Bedford Associates, was dedicated to developing, manufacturing, selling and servicing PLCs.[5]

A PLC consists of five main parts: CPU, Memory, Power Supply, I/O, and Communication port, which are shown in Figure 2.

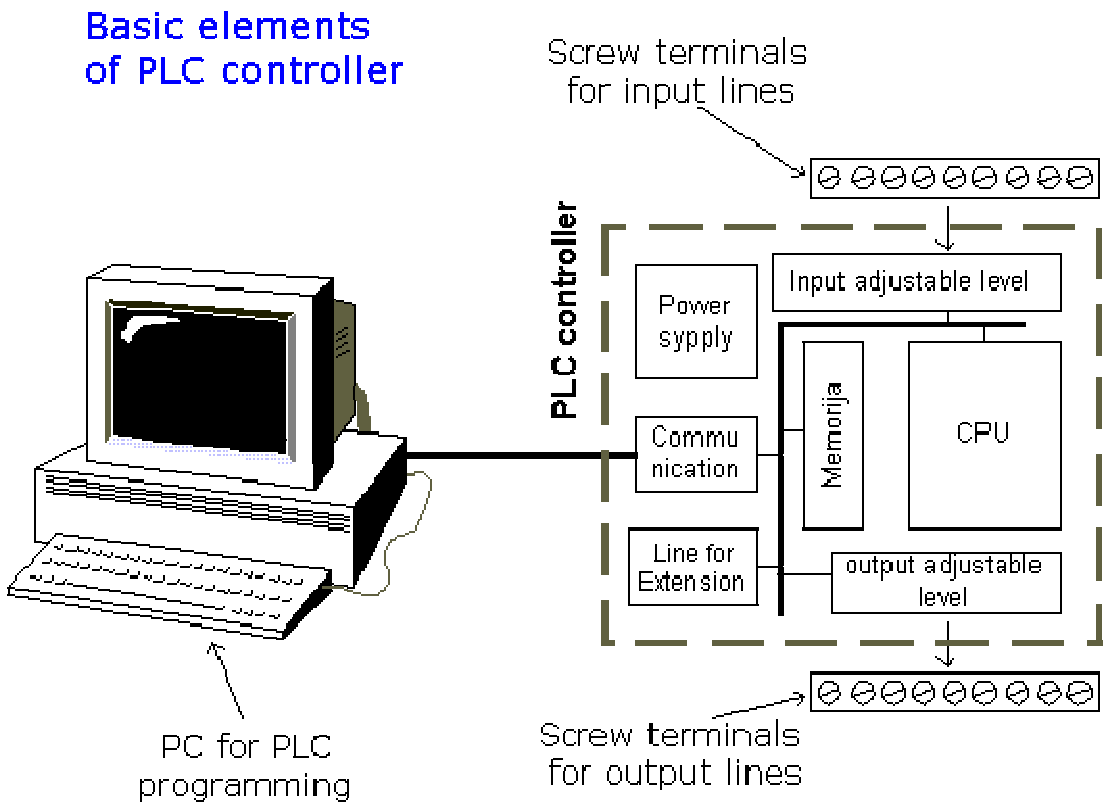


Figure 2: PLC hardware structure[7]

The first PLC programming language was “Subroutine Logic”, which is similar to electrical schematic diagrams. This language was created to reduce training of the existing electrical engineers. “Instruction List”, based on stack-based logic solver, was one of the early PLC programming languages, as well[5]. Today, a variety of PLC languages exist which can be used by an electrical engineer. All of the PLC languages are defined by the IEC 61131-3 Standard. This standard is published by the International Electrotechnical Commission (IEC). The IEC 61131-3 standard defines three graphical and two textual PLC programming languages[5]:

- Subroutine Logic (LD) – graphical
- Function Block Diagram (FBD) – graphical
- Sequential Function Chart (SFC) – graphical

- Instruction List (IL) – text
- Structured Text (ST) – text

Currently PLCs are used in a range of industries: Automotive, Dairy, Abattoir, Water Treatment, just to name few.

2.2 Literature Review

As the need for lower cost and better quality of projects is becoming increasingly vital for business survival, any opportunity for reduction of project cost is being explored[2]. One opportunity for the cost decrease is enabling automatic generation of databases and programs. As programs are becoming more complicated, so are their databases. The need to automate program and database generation, is becoming more apparent [8].

A blueprint for designing a software agent for automatic generation of a PLC program has been researched and documented. These research papers signify primary parts of a PLC program [1]:

- Nominal Sequence
- Alarms
- Start up
- Asset Monitoring
- Set up
- Manual Mode
- Diagnosis
- Shutdown
- Communication
- Safety functions

Automating PLC code generation has already been tried, and partly accomplished in different ways. The most established approach, for cost reduction in programming, is the re-usability of a developed code [1]. This approach can be considered as the first step in the automatic code generation cycle. This approach has been enhanced by the use of a Task-Oriented Method [9]

or, in RSLogix5000 language, by the use of Add-On-Instructions [10]. While this method is effective in standardizing parts of the code, it is not a true automatic generation of the code. It does save programming costs and time[10]. The main purpose of an Add-On-Instruction is to standardize a part of the code, concerning a particular device, such as motor structure, which ideally does not change [10].

The Task-Oriented Method was further enhanced, by creating a high-level language, which uses Task-Oriented blocks. In this approach, intention was to create a high-level language program for the operators, rather than the programmers. While this is an efficient way of saving costs, since operators are doing the programming rather than the programmers, it has many limitations. Only predefined codes can be used and only with the predefined parameters [11].

The approaches to cost savings in programming, mentioned above, initially require PLC programming, and thus these approaches cannot be considered as automatic code generation advances.

A more advanced approach to automatic code generation has been introduced by simulating the process flow, and from this simulation, a PLC code is created [1]. This particular approach is considered as an automatic code generation system, but it does have its limits. In this approach, only PLC sequential logic is created, and only for perfect world situations. However, the body of the PLC code is not created and thus some of the machine safety interlocks are not included in the final code [1] [12]. Another limitation of this approach is that the simulation packages are designed for one area of the particular industry type, such as an assembly cell of the car manufacturer [9] [12]. Because of these limitations, these simulation packages are not suitable for all control applications and a complete PLC code is not generated, even for the applications that the simulation package is designed.

As the approach above is used to create a PLC sequence code, there is also an approach to create the Body of the PLC code. As the approach above, this one also has its limitations. When the body code is created, the sequence (automatic mode) code is not created [13].

At present, only one product seems to be a truly Automatic PLC code generation program. It automatically creates a code for sequences as well as for the body part of the code [14].

(NOTE: An attempt was made by the author of this thesis to contact the creators of the program mentioned above, unfortunately, all the contact details were either incorrect or no longer in use.)

From author's personal experience and communication with electrical engineers from other companies, it is apparent that other forms of automated PLC programming exist. This information is classified as company's secret and as such, the information is not known outside the individual company. Consequently, it is difficult to assess the full extent of research done in this area.

2.3 Research Question and Hypothesis' of the Thesis

Based on the published literature it can be seen that the automatic generation of a fully functional PLC code is not widely available. A form of automatic PLC code generation does exists, but not to the extent of that detailed in the articles which outline what a PLC code generator should accomplish.

A key research question of this thesis is:

Can an APCG software be designed which generates a fully functional PLC code which will minimize the PLC programming time and costs?

A fully functional PLC code includes all the items stated in the research paper "Automatic generation of PLC code beyond the nominal sequence" written by Guttel et al [1] as well as in the standard of Realcold Milmech Pty. Ltd. (explained in the section 3.1).

It has been hypothesized that the following can be achieved:

- The APCG software can be created by using MS Excel and Visual Basic for Applications.
- The APCG software will create a fully functional PLC code.

- The APCG software will save time in designing, programming, and testing of a PLC code.
- The PLC code, generated by the APCG software, will become more reliable, than the PLC code created by traditional tools.
- Engineering Costs will be improved, when using the APCG software. With reduced engineering costs, a project's profit margin will be improved.

The APCG software will generate a PLC code for the Rockwell Automation's RSLogix5000 software. It will not work for other PLC programs such as Siemen's Step 7 software. As there are numerous PLC processors that use RSLogix5000 software, the generated PLC program will have Rockwell Emulator as a PLC processor [15].

The APCG software will generate a PLC code according to what is entered by the user. Functionality of the PLC code generated by the APCG software is outside of the scope of this project, and in turn, outside of capabilities of the APCG software.

2.4 Summary

The PLC was created in 1968, by Bedford Associates. The PLC was created for the American automotive manufacturing industry. It replaced expensive and time-consuming hard-wired control systems. A PLC consists of five main parts: CPU, Memory, Power Supply, I/O, and Communication port. Currently the IEC 61131-3 standard defines five PLC programming languages.

The need to automate PLC programming has been recognized as one of key areas that can lead to improved efficiency and enhance profit margin [1]. The method and structure of the automatically generated PLC code has been researched and documented [13]. The automation of the PLC programming has not been fully achieved according to the published research [1].

This thesis will attempt to create a software medium, the APCG software, which will automatically generate a PLC code according to the standard of the research paper by Guttel et al [1] and the PLC standard of Realcold Milmech Pty. Ltd. This thesis will also serve to

demonstrate the practical applicability of the APCG software and its ability to improve efficiency and the profit margin of a project.

3 Method

3.1 PLC Programming Structure

A good PLC code is written in a structured manner[16] and this is enforced in the Realcold Milmech Pty. Ltd. A structured PLC code can be crudely divided into two parts, the Body and the Brain[17].

PLC structure described will be based on the PLC Structure of the Realcold Milmech. The breakdown of the two parts is indicated below.

The Body:	PLC Inputs Alarms Flags SCADA/HMI interface	} Part 1
	Motor structure Solenoid structure PLC Outputs	} Part 2
The Brain:	Sequences	

Figure 3 shows the flow of the PLC program. As the inputs are read by the PLC controller, the first part of the Body code gets inputs ready for the Sequence. According to those inputs, the sequence sorts out which outputs should be turned ON and which outputs should be turned OFF. Once that is sorted, the sequence, using the second part of the Body, turns the PLC outputs ON or OFF.

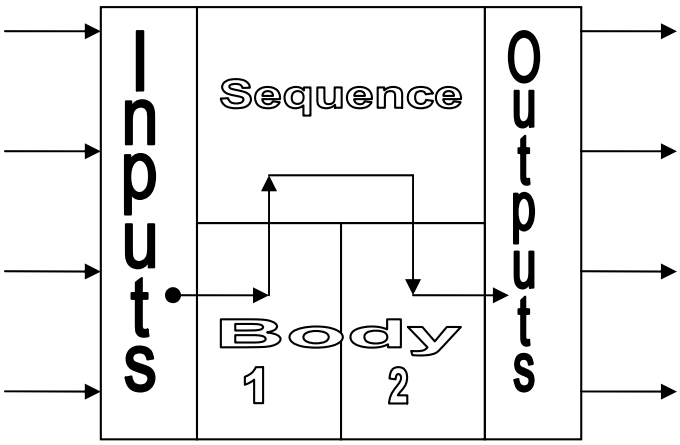


Figure 3 - Flow of the PLC program

3.1.1 PLC Program - The Body

According to the standard of the Realcold Milmech Pty. Ltd., each part of the PLC code is structured. There is a structure for the inputs, Alarms, Motors, Solenoids, Outputs, etc. The Sequence is done as a flow chart, in the PLC Subroutine language. Task-Oriented Method will be utilised for structuring Inputs, Motors and solenoids. The Task-Oriented Method in RSLogix5000 comes in the form of an Add-On-Instruction (AOI) and User-Defined-Data-Type (UDT).

Structure of Digital Inputs

Sensors, which are used as Digital Inputs, can be wired as Normally Open (NO) or Normally Closed (NC) inputs. Sensors wired as NC inputs are wired in “Fail Safe” mode. For NC inputs power is applied to the input while in good state. If the sensor operates (goes into the bad state) or the wire is broken than the power is gone from the input. Figure 4 show inputs wired as NC and NO. The input “LP Separator High Level Switch” is wired as NC input.

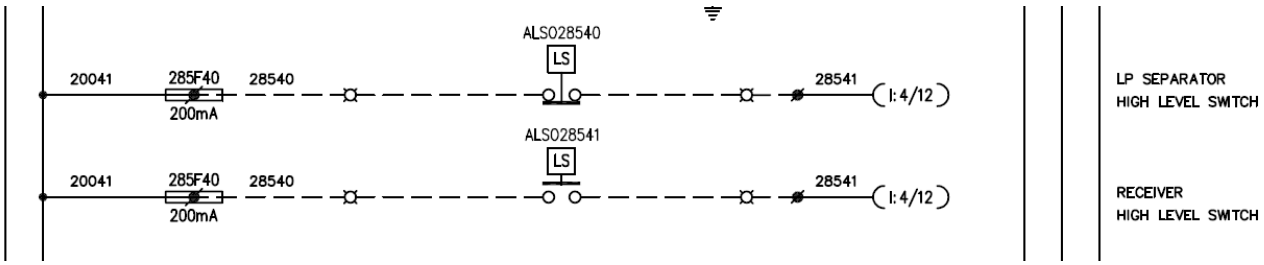


Figure 4 - Digital Inputs – wired as NC and NO

Each digital input is conditioned before the input is used by the internal PLC code. The conditioning is done by connecting the input straight to the PLC code (Figure 5) or by using a timer (Figure 6). Using a timer is important when there is a possibility of input flickering (quickly turning ON and OFF). This can occur in applications where photoelectric sensors are used, such as on conveyor applications. Usually 0.5s timer delay is enough to secure correct digital input.

(Note: In RSLogix5000, the time base for timers is 1 microsecond. 0.5s = 500 μs, as shown in Figure 6)

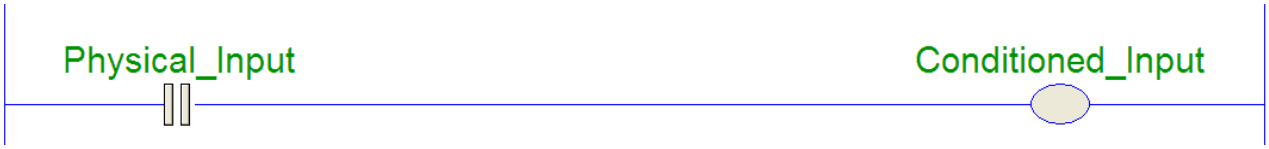


Figure 5 - Digital Input Conditioning – Straight through

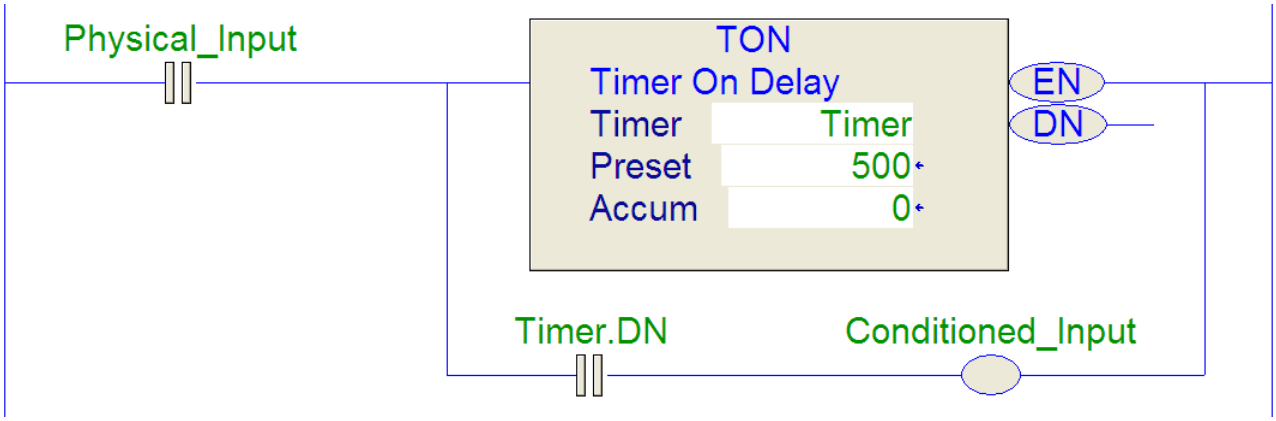


Figure 6 - Digital Input Conditioning – Using a timer

Alarms are one of the most important parts of the PLC code, especially in the machine programming, where there are moving machine parts, which can harm an operator or maintenance personnel. Since there are Digital and Analogue inputs, Digital and Analogue alarms are also required. Digital alarms use a timer similar to the Digital input conditioning (Figure 6).

There are two types of the Digital Alarms.

1. Indication Alarm
2. Cutout Alarm

Indication Alarms are used for indication purposes, only. These alarms do not stop any processes but merely inform the operator that there could be a potential problem. These alarms come on when in the alarm state, stay on while in the alarm state, and disappear when the alarm goes away (Figure 7).

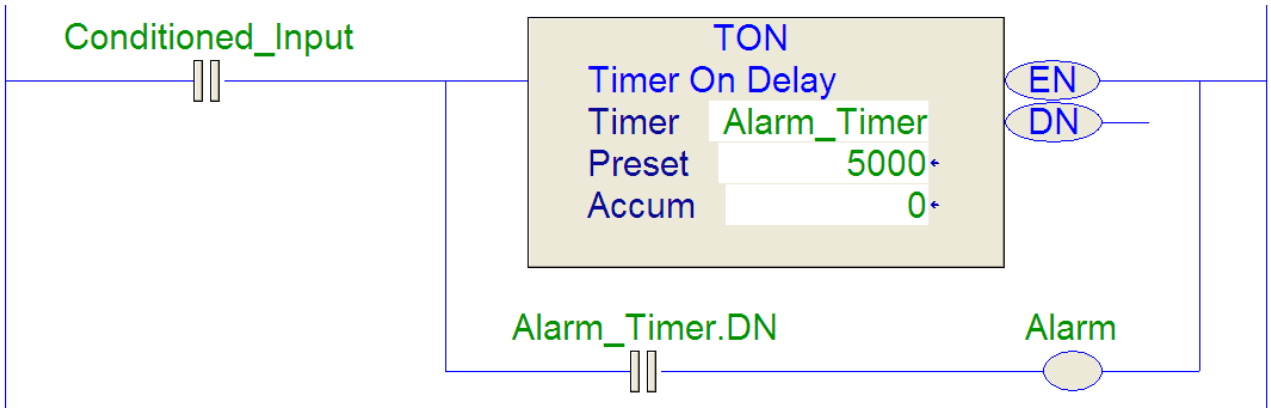


Figure 7 - Digital Indication Alarm

The Cutout Alarms stop a process, a machine, or a part of a machine. This type of an alarm stays on until the alarm is gone and operator resets the alarm (Figure 8). The Cutout alarm MUST not reset itself. An operator MUST reset the Cutout Alarm. To keep this alarm ON even though the alarm is gone, an input is paralleled with the alarm, and a Reset button as shown in Figure 8, this is called “Self Latching”. When “Conditioned_Input” comes on after 5 seconds an “Alarm” bit will be turned ON. Parallel to the “Conditioned_Input”, an “Alarm” bit and normally closed “Reset” bit, create another path for an alarm to stay on after “Conditioned_Input” is not ON anymore. Once an operator presses “Reset” button the parallel path will be cut and the alarm will turn OFF, if “Conditioned_Input” is OFF.

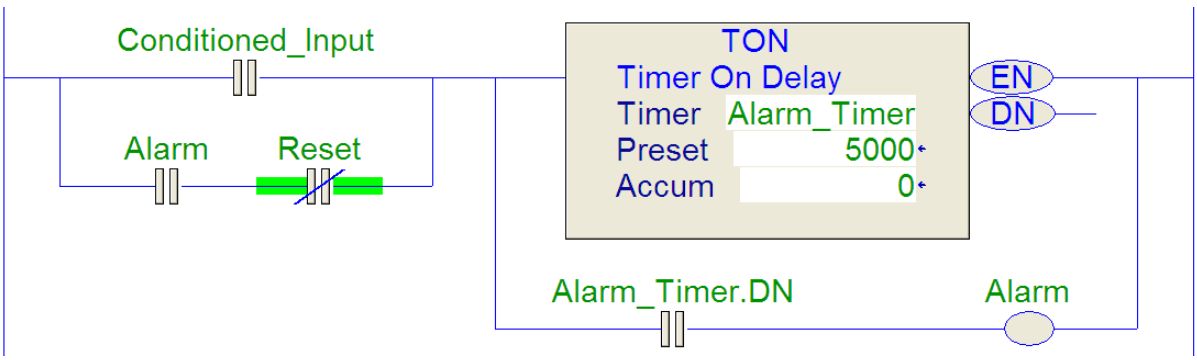


Figure 8 - Digital Alarm – Cutout

The structure, of Digital Inputs, does not change as different inputs are programmed. As it is required to minimize programming time but increase the programming quality an Add-On-Instruction (AOI) can be utilized. The AOI for Digital Inputs has two parts: User Defined Data Type (UDT) and Subroutine Logic program. The UDT needs to have two bits, Conditioned Input and Alarm, and two timers, Conditioned Input and Alarm Timer. The Subroutine Logic would contain four rungs as shown in Figure 5, Figure 6, Figure 7 and Figure 8.

AOI logic will be slightly different than what is explained above. In Figure 8 only an input is considered for an alarm. In reality, this is not always the case, and other variables might be part of a condition, which must be met before an alarm is raised. An AOI is a Task-Oriented-Method, and the structure does not change as inputs are programmed in, so an interlock must be put in the condition for an alarm to be raised. This interlock represents all other conditions, which are required for an alarm. The interlock needs to be one of the variables of the Digital input AOI instruction. In Figure 9 an interlock ("Alarm_Interlock") is put in series with the input ("Conditioned_Input"), so both of the conditions must be met before an alarm is raised.

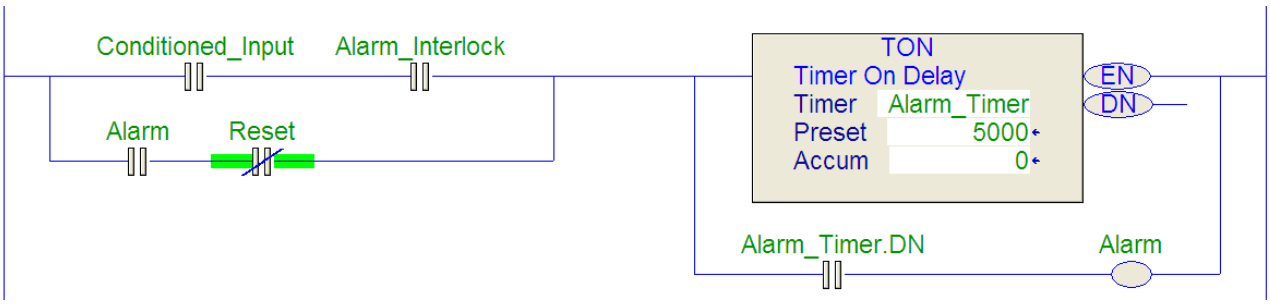


Figure 9 - Digital Alarm - AOI Interlock

Structure of Analogue Inputs

Analogue inputs need to be conditioned for the PLC code, as well. The conditioning comes in the form of the "*scaling*" function.

A scaling function has two ranges a Physical (Raw) Input and Engineering Units range. Raw Input range depends on the PLC input card and Sensor type. Analogue Input is divided into two types, Voltage Input and Current Input.[18]

1. Voltage Input
    - -10-10 V dc
    - 0-10 V dc
    - 0-5 V dc
    - 1-5 V dc
2. Current Input
    - 0-20 mA
    - 4-20 mA

Analogue input type, used by Realcold Milmech Pty. Ltd., is current input 4-20 mA. In the example below, 4-20 mA analogue input will be used.
Engineering Input range is an Engineering unit representation of the Physical Input range. The Engineering unit representation is operator entered. The scale function has six parameters:

1. Raw Input (In)
2. Raw Input Minimum range (RawMin)
3. Raw Input Maximum range (RawMax)
4. Engineering Unit minimum range (EngMin)
5. Engineering Unit maximum range (EngMax)
6. Offset value (Offset)
7. Scaled Output (Out)

In Figure 10 a “Scale” function, used by RSLogix5000, is shown. This function takes in a Raw analogue value, 4-20 mA. The Raw analogue value is scaled against an Engineering unit range, 0-100. Engineering Unit can be anything, but for the example used below, a percentage “%” will be an Engineering unit.

Offset parameter, shown in the Figure 10, is used if the sensor calibration is not correct. This parameter should be used only temporarily, until the sensor can be recalibrated or replaced. Output parameter (value 50%) is the result of the “Scale” function.

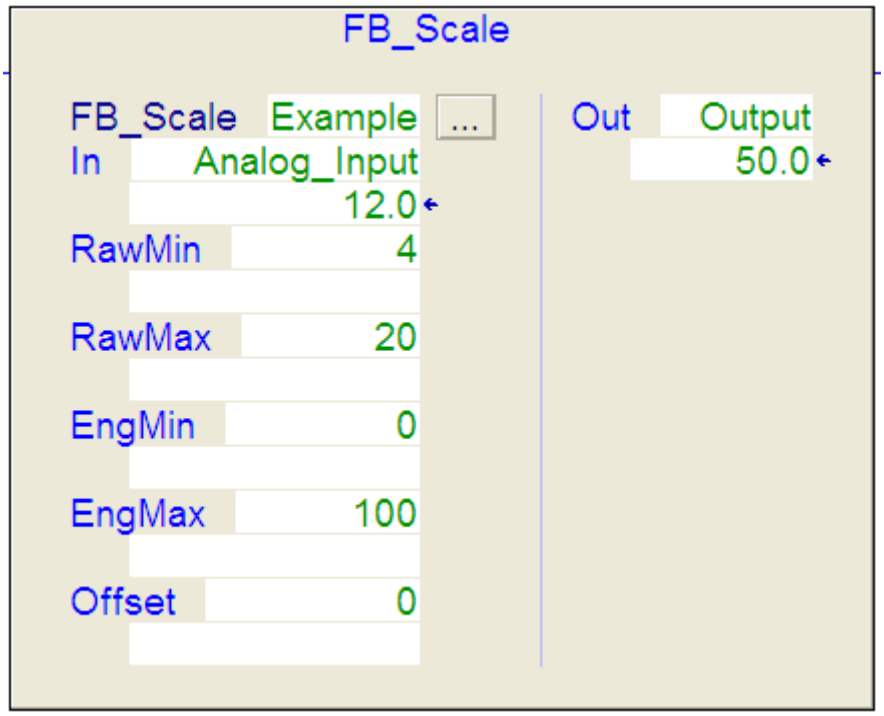


Figure 10 - Analogue Input Scale function

The equation the “Scale” function is:

$$\text{Out} = \frac{\text{In - RawMin}}{\text{RawMax - RawMin}} \text{ x (EngMax - EngMin) + EngMin + Offset}$$

Example from Figure 10:

$$\text{Out} = \frac{12 - 4}{20 - 4} \text{ x } (100 - 0) + 0 + 0$$

$$= \frac{8}{16} \text{ x } 100$$

$$= 50$$

Analogue Alarms differ from Digital Alarms. The main difference is that Analogue Alarms have more than one alarm per analogue input. There are as many as five analogue alarms per analogue input:

1. High Cutout
2. High Alarm
3. Low Alarm
4. Low Cutout
5. Sensor Failure Alarm

Not all of these alarms are used (monitored) by the PLC program. But these alarms are available to be used if needed [17]. The analogue alarms do follow the structure of the digital alarms. Cutout and Sensor Failure alarms are used to stop the process while other alarms are used as an indication only. Cutout alarm is a "Self Latching" alarm, just like in Digital Cutout Alarms.

Figure 11 shows an example of an Analogue Cutout Alarm. In the example, if a Conditioned Analogue Input (value 50) is greater than the Cutout Alarm Set Point (value 40) than an alarm is activated after 5 seconds. As this is a Cutout alarm, this alarm is "Self Latching", which means that even if the Analogue Input value goes below the Set Point, an alarm will still stay on until the operator sees and resets the alarm, by pressing "Reset" button.

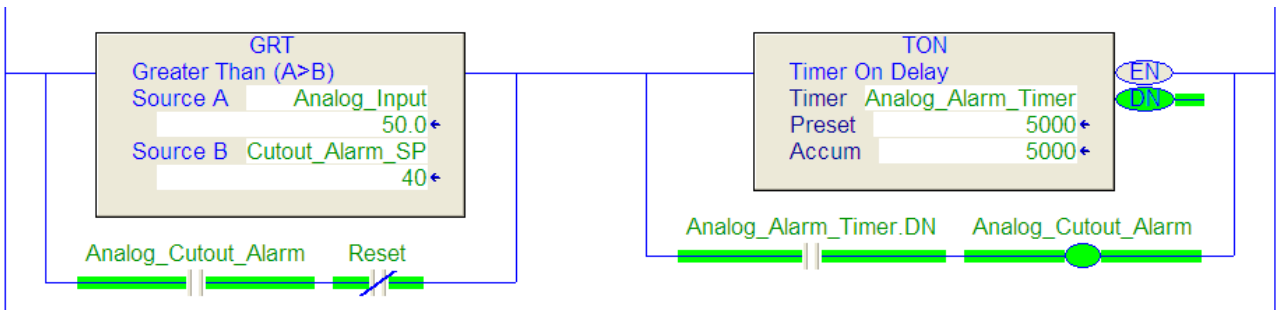


Figure 11 - Analogue Cutout Alarm

Because the Analogue sensor has a Physical (Raw) Input range, analogue input can go below or above the range. There are three reasons why this can occur:

- The sensor is disconnected. A fuse might have been blown.

- The sensor is faulty. Moisture or noise can cause this.
- The selected sensor's range is not appropriate. As an example, if sensor's range is 0 – 900 kPa, but in real world the range is -100 – 900 kPa, than when the pressure goes below 0 kPa sensor will be outside its sensing range, and thus the Analogue Input will go below Physical Input range, which is 4 mA. It is assumed that the engineer who has selected sensor's range has done the job appropriately, so initially this is assumed not to be one of the problems.

Because of the reasons mentioned above, a sensor failure alarm is also added to analogue alarms associated with each analogue input.

As it was done for Digital Inputs, where an AOI was made, which incorporates Input conditioning with the Alarm conditioning, Analogue Inputs AOI was also created. The structure of the Analogue sensors does not change as different analogue inputs are programmed.

This AOI contains all the attributes of the Analogue input conditioning (scaling) and Analogue input alarms. A UDT has been created for this function. The UDT for analogue inputs, is bigger than the UDT for digital inputs. The created UDT database holds 22 parameters for each analogue alarm:

Scaling (7 parameters)

- Raw Input (In)
- Raw Input Minimum range (RawMin)
- Raw Input Maximum range (RawMax)
- Engineering Unit minimum range (EngMin)
- Engineering Unit maximum range (EngMax)
- Offset
- Output

Alarms (15 Parameters)

- Low and High Cutout Alarm Bits
- Low and High Cutout Alarm Set Points
- Low and High Cutout Alarm Timers

- Low and High Alarm Bits
- Low and High Alarm Set Points
- Low and High Alarm Timers
- Sensor Failure Alarm Bit
- Sensor Failure Alarm Set Point
- Sensor Failure Alarm Timer

As it was done for Digital Input Alarm in AOI instruction, each alarm in the AOI will require to have an interlock associated with it. This interlock will have the same function, which it has in the Digital Input Alarm.

Flags

The "Flag" subroutine is used for simplifying the PLC code, specifically, for the Sequence part of the PLC code. Sequence is done in step format (this will be discussed in detail in the sequence section of this chapter). The PLC code goes from, one step to the next, if the transition conditions are satisfied, as shown in Figure 12. Transition logic can be very extensive especially in moving machines where there are many conditions that need to be met before a machine can move (an example is shown in Figure 13). Instead of writing many conditions in-between steps, it is "cleaner" to group conditions, and to create a flag to represent grouped conditions.

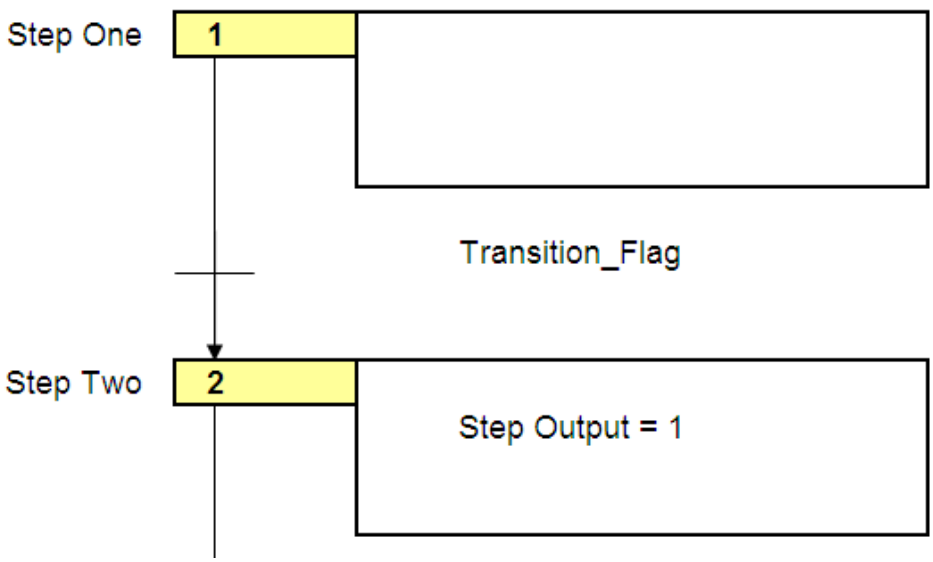


Figure 12 - Sequence Step Flow

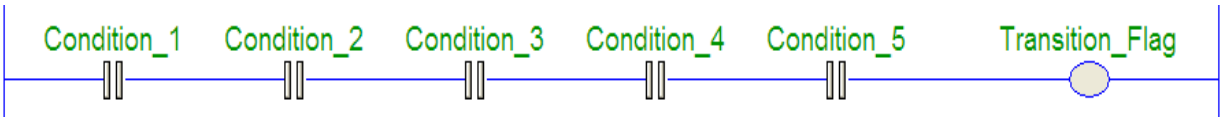


Figure 13 - Transition Flag

The “Flag” subroutine is used, as outlined by the research paper by Guttel et al [1], for:

- Start up flags
- Shutdown flags
- Monitoring
- Diagnosis
- Mathematical calculations

SCADA/HMI Subroutine

This subroutine, as the name suggest, is used for combining all of the tags that are used by the PLC code as well as “SCADA/HMI” program. The major purpose for this subroutine is to control the operators input, into the PLC’s Set Points. The Operator can enter a value into a Set Point which can potentially hold a value that will confuse the machine or a process and will, in turn, make a machine or processes do some odd if not dangerous things. Therefore limiting the value which operator can enter is very important for designing a reliable and “fault-proof” PLC code.

While writing to a Set Point is one directional, from SCADA or HMI to the PLC tag, there are, also, an indication tags (registers) which write PLC information to the SCADA or HMI. These tags (registers) are used by the operator to see what is happening with the machine i.e. Motor Running/Faulted, Solenoid Opened/Closed, etc.

Motor Structure

Every motor has its own structure, and the structure is the same for each motor. The motor structure contains Machine Interlock, Start Permissive, Start Command and Start Fault Alarm.
In the very least, each motor has one input (running indication) and one output (start) associated with it.

Machine Interlock and Start Permissive are safety parts of the Motor Structure. Machine Interlock is used to stop a motor from running if any of its conditions are not met prior to or

while the motor is running. This item is very important for operator and machine safety. All of the major safety items are put into this rung, as one of the interlock conditions i.e. Emergency Stop Push Button.

Start Permissive is process safety. It is important only prior to the start of the motor. Once the motor has started, the conditions in the Start Permissive are bypassed. Any conditions that need to be taken into account, only prior to the motor starting should be put into this rung.

To explain this concept, an example will be used, shown in Figure 14. A pusher is used to push a box onto a conveyor. Once the box is on the conveyor, the conveyor takes the box away. Pusher can only push a new box onto the conveyor if the conveyor is empty, that is, a previous box has left the space where the new box is to be pushed on. A sensor is used to indicate if there is a box on the conveyor. Once the conveyor is empty (Step 1), and the pusher is pushing a new box onto the conveyor (Step 2), initially the sensor will see that there is no box on the conveyor. But as the box is being pushed on the conveyor the box will be sensed by the sensor even though the box might not be completely pushed into the position (Step 2). Once the box is completely pushed on the conveyor (Step 3), pusher will retract and conveyor will move the box away (Step 4). Pusher cannot push another box onto a conveyor until the conveyor takes the first box away (Step 5).

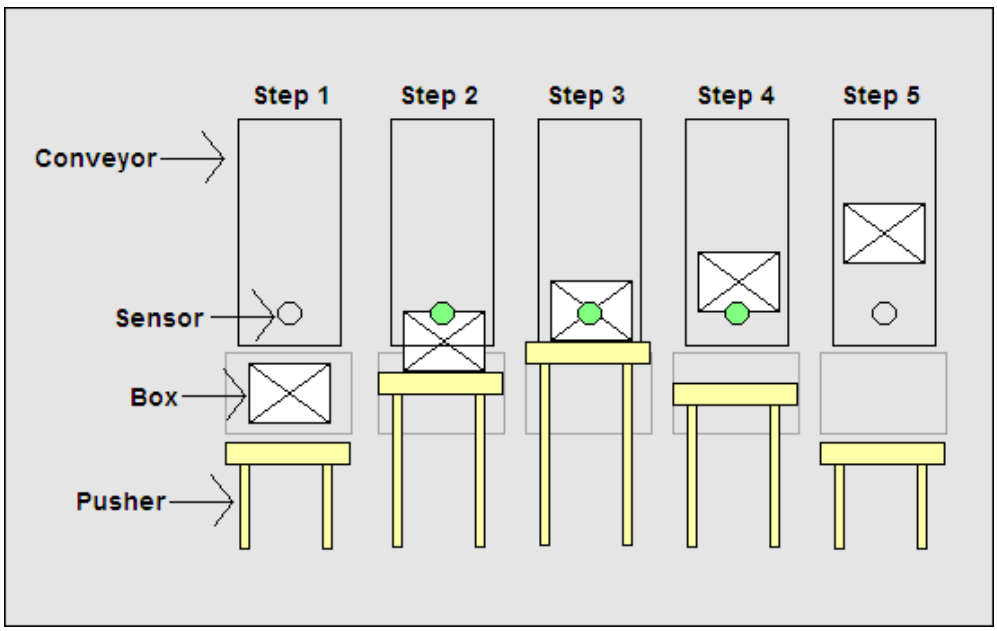


Figure 14 - Start Permissive (Example)

In the example shown in the Figure 14, Start Permissive would be used in Step 1. Once the pusher starts, “Start Permissive” is bypassed and not taken into account until Step 5. Figure 15 shows the PLC code for “Start Permissive”. If Box is not present (Box_Present_Sensor) “Start Permissive” is energized, but once the Pusher is pushing (Pusher_Pushing), the sensor is bypassed until the Pusher stops pushing (Step 4 in Figure 14)

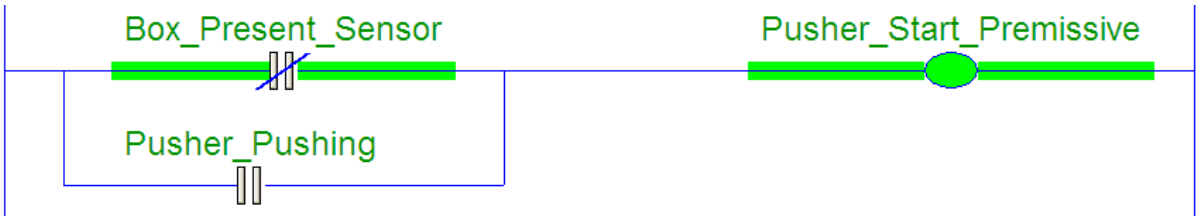


Figure 15 - Start Permissive (PLC code)

For the motor to start, the conditions mentioned above need to be met, but also the motor has to be put into a Manual or Automatic state. If the motor is in the Manual state, it will run as soon as the Motor Interlock and the Start Permissive conditions are met. However, if the motor is in the Automatic state than the motor will start when the sequence (the brain) tells the motor to run if the Machine Interlock and the Start Permissive conditions are met.

In Figure 16, the motor has been asked to start (Motor_Start). The motor is in the Automatic state (Automatic_State) and the sequence is telling the motor to start (Sequence_Output). The Machine Interlock and the Start Permissive conditions have been met.

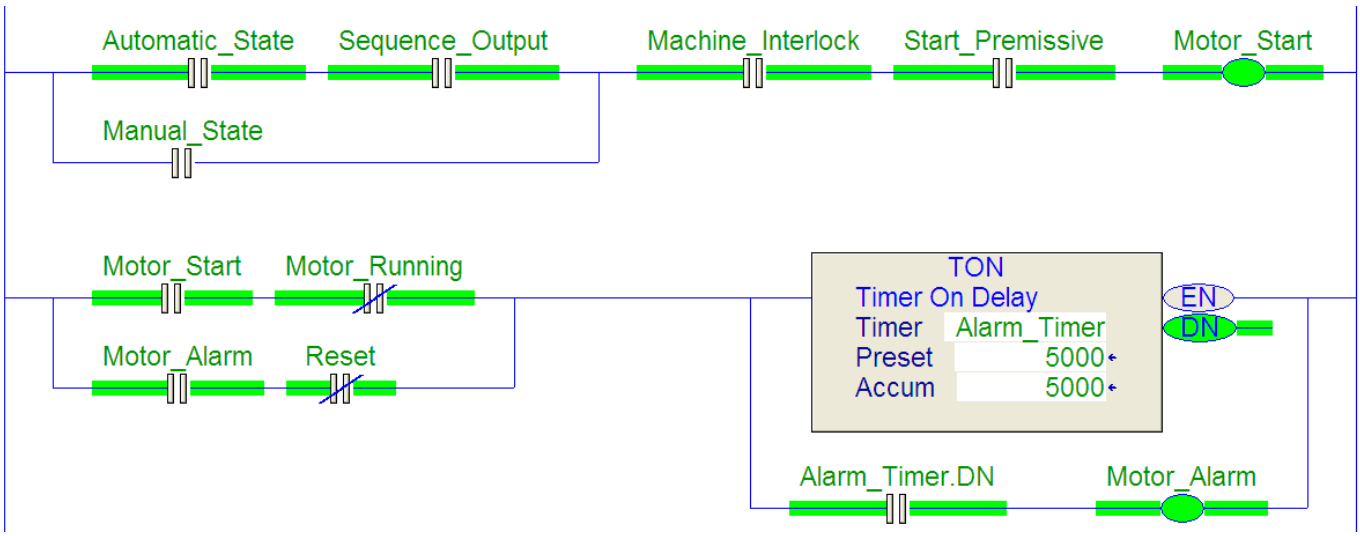


Figure 16 - Motor Start condition

“Motor Start Alarm” occurs when the motor is asked to run but it does not. There are a few reasons why this can occur: overload trip, loose wire, motor isolator is in off position, etc. When this situation occurs, an alarm is raised. The “Motor Start Alarm” is a self-latching alarm, and an operator needs to reset this alarm, before the motor will try to start again. Figure 16 (second rung) shows the structure of the “Motor Start Alarm”. In the figure, motor has been asked to run (Motor_Start) but the motor is not running (Motor_Running), after 5 seconds alarm has been raised (Motor_Alarm). Because this is a “Self Latching” alarm, the alarm bit is paralleled with the alarm condition.

The structure above is only a “base” structure, and more extensive motor control structure can be programmed in if necessary. This base structure is used for majority of motors. Other motor controls that can be included are:

- Motor Running Reverse
- Motor Start Reverse
- Motor InService – This can be used when other parameters are important for the motor to start, but those parameters do not belong in Machine Interlock and Start Permissive.

The motor structure explained above does not change, from motor to motor. An AOI can be used to atomize and standardize, even further, a motor structure. Only, “Motor Start” and “Motor Start Alarm” rungs can be put into an AOI, since these two items do not change at all. “Machine Interlock” and “Start Permissive” may change, depending on the function of the motor. Because of that, the Machine Interlock and the Start Permissive will have to be external attributes to the AOI. A UDT, which accompanies the AOI, is also created, and the database will have 12 parameters:

- Manual Mode
- Automatic Mode
- Sequence Start
- Machine Interlock
- Start Permissive
- Motor Running
- Motor Running Reverse (not used very often)

- Motor Start Alarm
- Motor In Service (not used very often)
- Motor Start Command
- Motor Start Reverse Command (not used very often)
- Motor Start Alarm Timer

Solenoid Structure

A Solenoid structure is the same as the Motor Structure. The functionality is the same. Solenoid parameters are:

- Manual Mode
- Automatic Mode
- Sequence Start
- Machine Interlock
- Start Permissive
- Solenoid Opened (not used very often)
- Solenoid Closed (not used very often)
- Solenoid Open/Closed Alarm (not used very often)
- Solenoid in Service (not used very often)
- Solenoid Open Command
- Solenoid Close Command (not used very often)
- Solenoid Alarm Timer (not used very often)

As it was done for the Motors, Solenoids have their own AOI instruction. As it was done with the Motor AOI instruction, the “Machine Interlock” and the “Start Permissive” are coded externally to the AOI instruction. Because some of the Solenoid’s parameters are seldom used, they will not be part of the AOI. The rung, which controls “Solenoid Open Command”, would be the only rung in the AOI instruction. The fact that there is only one rung in the AOI instruction might raise few questions:

1. Is it necessary to have an AOI instruction?

2. Will the AOI instruction save time?

The simple answer to these questions is NO. It is not necessary to have an AOI instruction and probably it does not save time. However, Solenoid AOI instruction has been made and the reasons for this are:

- Any future changes, add-ons, to the structure, are simple to do.
- It is always good practice to keep things in the same order, because it creates less confusion and fewer problems.
- It will be easier for the APCG software to use the same structure.

Digital Outputs Structure

As was the case with the PLC Inputs, there are Digital and Analogue PLC Outputs. There is no Digital Outputs conditioning. All the conditioning is done internally, and when all the conditions are met, the output is turned ON or turned OFF.

Analogue Outputs

Analogue Outputs do have conditioning, similar to the Analogue Inputs, but in reverse. It is necessary to convert the Analogue Output, which is within Engineering Units range into a Physical Output Range. Just like Analogue Inputs, Analogue Outputs are divided into two types Voltage Output and Current Output.[18] A "Scale" function is used for the conditioning. The Analogue Output type, 4 – 20 mA is preferred output type used at Realcold Milmech Pty. Ltd.

3.1.2 PLC Program - The Brain

Grafcet Functional Description

The PLC sequences are done according to the Grafcet functional description[17]. Each sequence has steps that it goes through, starting from step 0. In-between these steps, there are transitions, whose conditions must be met. When the conditions of a transition are met, than the sequence goes from one step to the next. Only one step can be active at any time. In any step an action can occur, or be stopped. In the initial steps, usually, there are no actions. They are mainly there to check all of the conditions prior to operating an actuator. In Figure 17, in step 1

there are no actions. Steps that are higher in the sequence, such as “Step 2”, in Figure 17, are used for starting (turning ON) an actuator.

As the steps are increasing, moving forward, there are also transitions that move steps backward (“Transition_2” shown in Figure 17). Each sequence needs to have a “return path”. Return path is used to return the sequence to the step prior the “action” step, a step in which an action does not occurs.

The most important feature in the sequence is a safety feature, which is moving back to the first step from any step. This usually occurs when an Emergency Stop is pressed or another Cutout alarm occurs. In Figure 17 when the “Emergency_Stop” is pressed the sequence returns to the “step 1”.

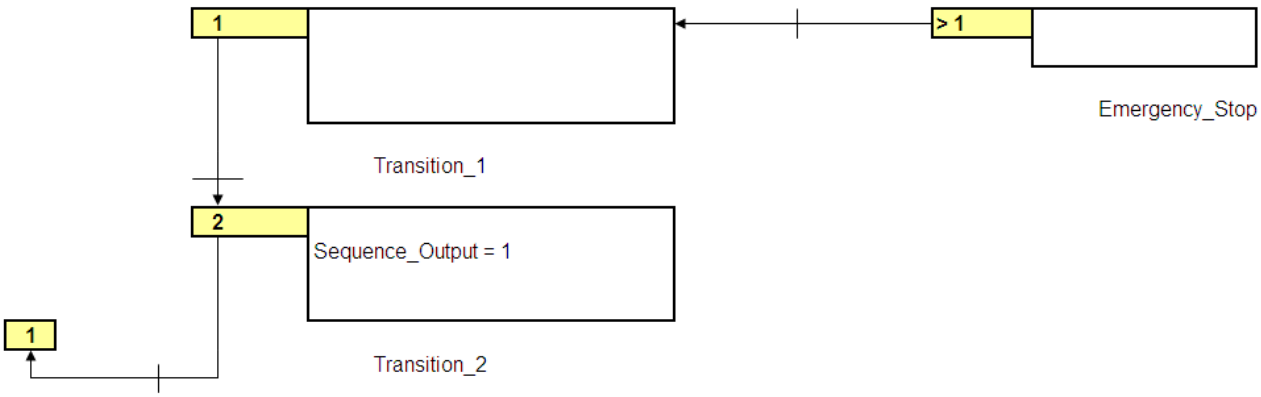


Figure 17 - Grafcet Steps

The sequence can become extensive and confusing, so an engineer should keep the sequence as simple as possible, by utilizing “Flags” subroutine, explained earlier in this chapter.

There are numerous ways of writing (drawing) a Grafcet Functional Description, and it tends to vary between companies.

PLC Sequence

Every project needs to be broken down into areas, and than those areas need to be broken down into devices. Engineers have their own way of writing a PLC program. Each engineer decides how many sequences there will be in a PLC program. As a rule, at Realcold Milmech Pty. Ltd.,

each actuator should have its own sequence; this keeps the PLC code simple and easy to understand.

PLC sequence has three major parts:

1. Sequence Flow
2. Sequence Steps Allocation
3. Sequence Action

Sequence Flow, as the name indicates, is the flow of the sequence. It has steps and transitions. If the sequence is at step 1, it looks at the transition condition, and if the conditions are met, it moves to the next step (Figure 17).

Figure 18 shows PLC representation of Figure 17. The Figure 18 shows that the current step is step 2, and waiting for "Transition_2" to be set, in which case the step would go back to step 1. In addition, it shows that if the "Emergency_Stop" is pressed, and the current step is not one, than the step will go back to one regardless, of where the step was prior to it.

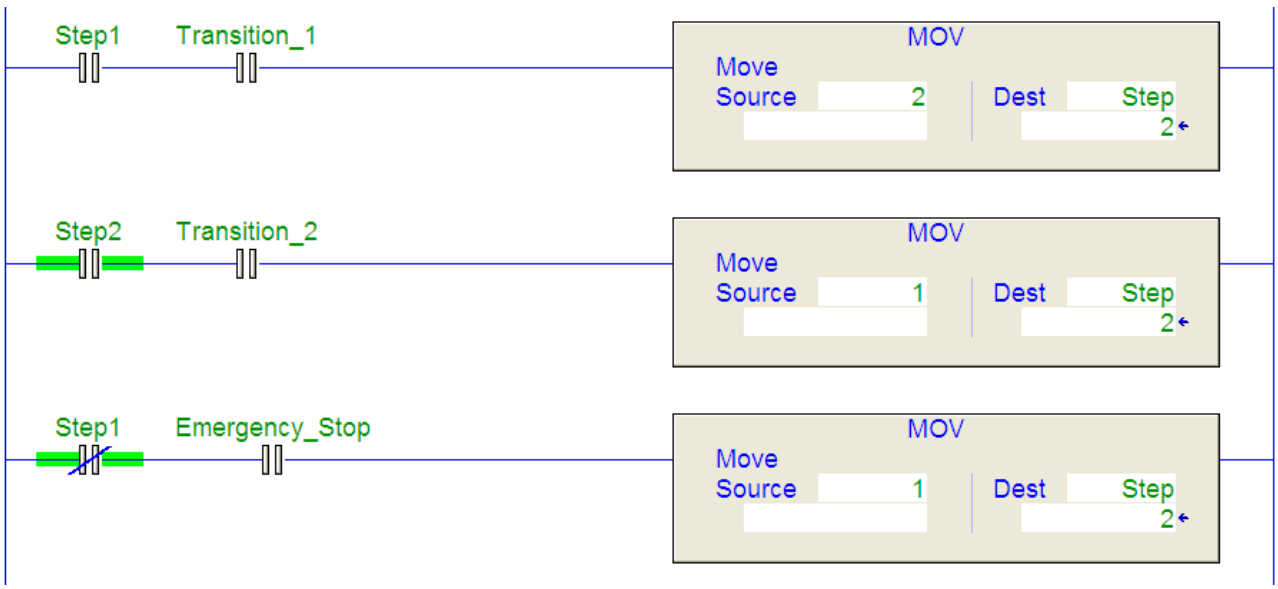


Figure 18 - PLC Sequence – Steps

It is worthy to note that current step is programmed on the left hand side of the rung, while the next possible step is programmed on the right hand side. The flow of the rung is:

Current Step → Transition → Next Step

Sequence Steps Allocation sorts out what the current step is. All of the Sequence Step bits turn ON or OFF in this part of the sequence (Figure 19). Each of the current steps (left hand side in Figure 18) needs to be turned ON and OFF at a proper time, which is the time when the sequence is in the step that corresponds to the Sequence bit.

Figure 19 shows the PLC representation of the Sequence Steps Allocation. In the figure, current step should be step 2, and the Sequence bit for step 2 is turned ON ("Step2").

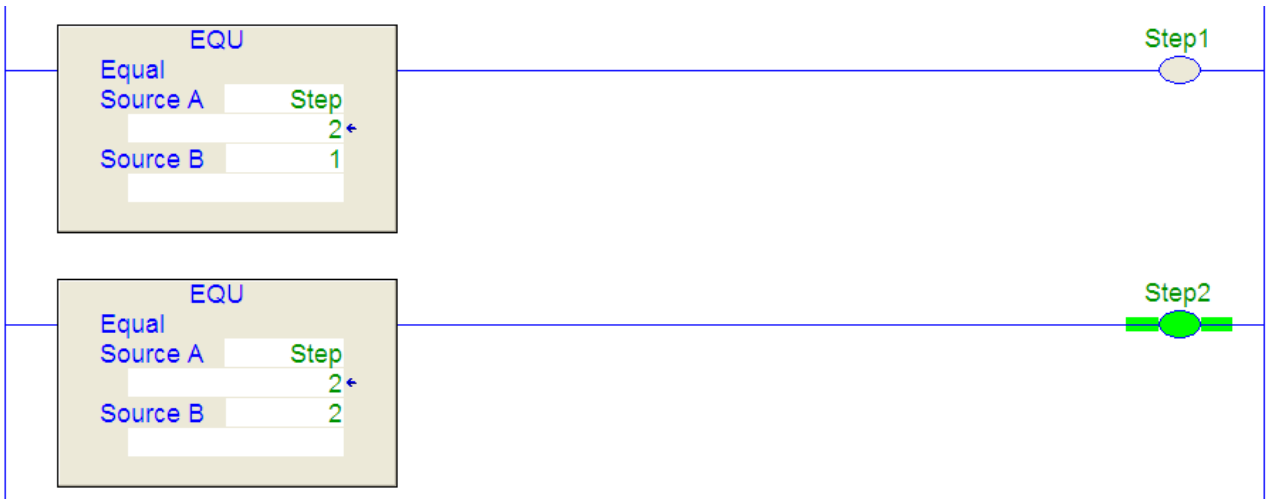


Figure 19 - Sequence Step Allocation

There are a number of ways of writing a sequence. An engineer may only use a Sequence Integer (in Figure 19, the integer is "Step") and no Sequence bits. Unfortunately, that way has a flaw, a "coding error". In one PLC scan, Sequence Step can go from step 1 to step 2, than to step 3, than to step 4, etc. (as an example). However, in the, Figure 18 and Figure 19 in one PLC scan Sequence step can only move one step at a time.

Sequence Action is where a device, which is in Automatic mode, is turned ON or OFF, if the interlock and the start permissive conditions are met. Figure 20 shows that in step 2 ("Step2) a sequence output is turned ON. This sequence output can be used for Motor, Solenoid or another actuator.

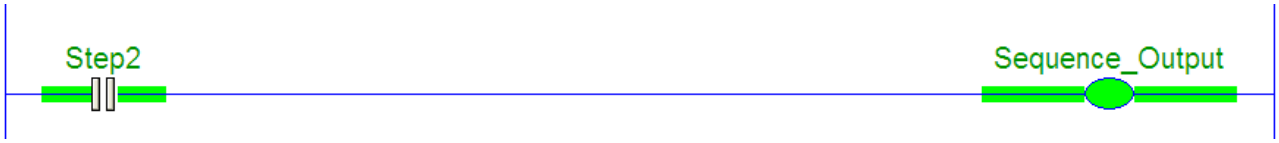


Figure 20 - Sequence Action

By looking at Figure 16 it can be seen how the sequence output is incorporated in the Motor structure.

3.2 RSLogics5000 Files

The Rockwell RSLogix5000 program has an option to be saved as an ACD file or as an L5K file. A PLC program in the ACD file is used by the Automation Engineers for programming. An L5K file can be programmed and modified in Notepad. The L5K file is not used for programming, but this option allows for other programs such as VBA to modify the L5K file.

The APCG software, when automatically creating a PLC code, will use an L5K file. This L5K file will be created in Notepad text format. An engineer will open an L5K in the RSLogix5000 program.

3.2.1 ACD File

To understand how to modify the L5K file, it is necessary to understand the parts that make up an L5K file. The easiest way of understanding an L5K file is to start from an empty RSLogix5000 program. To create an empty L5K file, an empty ACD file needs to be created, first. When creating a new RSLogix5000 file, a controller type must be selected (this is a requirement to create a new program). There are over 30 possible controller types, which can be selected. The APCG software will only work with RSLogix Emulate 5000 Controller. There are a few reasons for selecting the RSLogix Emulate 5000 Controller.

- Emulate Controller is designed so that it can be connected to the RSLogix Emulate 5000 software. It is a PC based program, which simulates a real PLC controller. Therefore, an engineer can download the program into the RSLogix Emulate 5000 software and test their PLC program.
- It is easy to convert an RSLogix Emulate 5000 Controller into any real PLC controller (i.e. CompactLogix5343 Controller – 1768-L43).

Figure 21 shows an empty RSLogix5000 program file. In the empty, ACD file there are no items, such as:

- User Defined Data Types (pointer 1 in Figure 21)
- Add-On-Instructions (pointer 1 in Figure 21)
- Tags in Controller Tags List (pointer 2 in Figure 21)
- Tags in Program Tags List (pointer 3 in Figure 21)
- Subroutine Programs (pointer 4 in Figure 21)

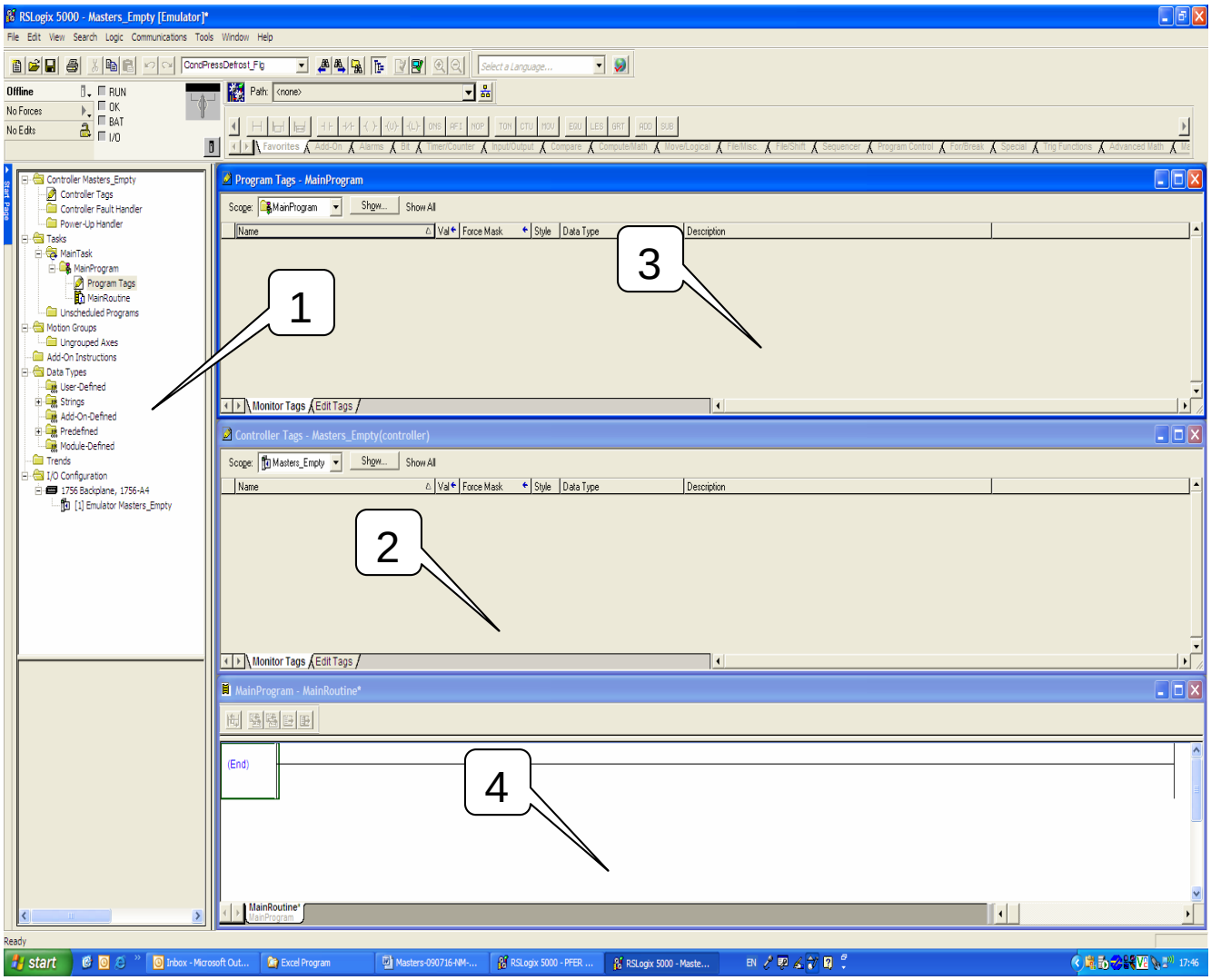


Figure 21 - Empty PLC program .ACD

3.2.2 L5k File

Once an empty ACD file is created, as shown in Figure 21, this file can be saved as an L5K file. When an empty L5K file is opened in Notepad, the L5K file contains the information about the

selected controller as well as the version of the RSLogix5000 program. This section only changes when a controller type is changed.

The layout of an L5K file is as follows:

- RSLogix5000 version
- Controller Type and Module
- Program Code
    - Controller Tags
    - Program Tags
    - Program Routines
- Controller Configuration

Controller Tags are accessible by all of the Programs, as there could be more than one program.
Program Tags are accessible only by the Program where the Program Tags reside.
Program Routines is the area where the program is written.

The tags lists (Controller and Program) expand as the number of tags grow. Both tag lists are structured the same, the only difference is the location of the list in the L5K file. The Figure 22 shows two tags in a Tag list, BOOL and an INT (Integer). Each tag list starts with the "TAG" heading and end with "END TAG" heading. Each tag is written in-between those two headings.

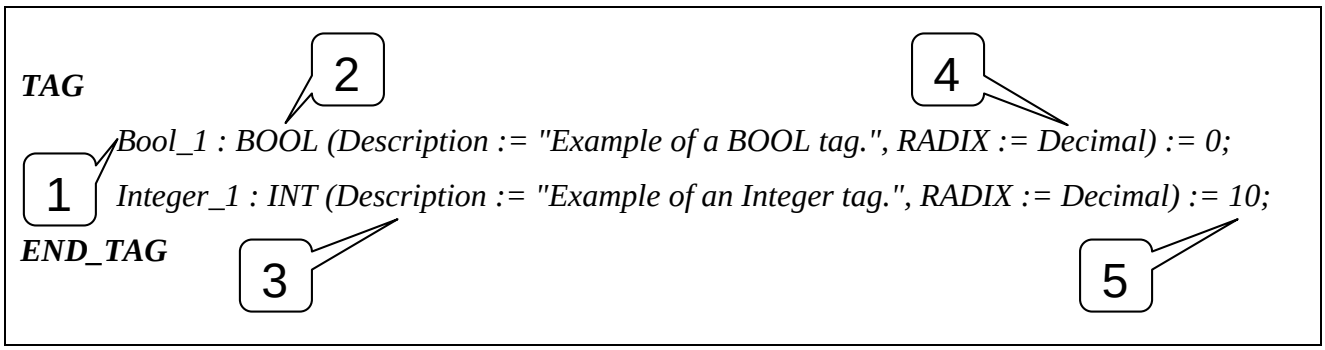


Figure 22 - L5K File Tags structure

The format of a tag is Name, Type, Description, and Value. Pointers in Figure 22 point to each tag part.

1. Name – Tag Name is the first part of a tag. It can hold up to 40 characters, which includes letters, numbers, and an underscore. Tag name cannot hold any other characters such as $, %, &, etc.
2. Type – There are various types of tags, but the most important once are BOOL, INT, DINT, REAL, TIMER AND COUNTER. Other important tag type is a UDT, which is a group of tags defined by a user.
3. Description – Describing a purpose of a tag is very valuable tool. In RSLogix5000, a tag description can hold 20 characters in one line, but it can have multiple numbers of lines. There are no restrictions on the character types, which can be used in the Description.
4. Style – This is a format of a displayed tag. The style used by an engineers in Realcold Milmech Pty. Ltd. is the Decimal format. There are six types of Style [19]:

Style	*Base*	*Notation*
• Binary	2	2#
• Decimal	10	
• Hex	16	16#
• Octal	8	8#
• Exponential		0.0000000e+000
• Float		0.0

5. Value – This is the initial value a tag holds. In Figure 22, Integer_1 tag holds value of 10.

After the Controller Tag list comes “Program” section. RSLogix5000 can hold multiple Programs. Each program contains a Program Tag list and Program Routines. A Program section starts with the heading, “PROGRAM”, and finishes with the heading, “END_PROGRAM”. The Program Tag List (explained above) is the first element of the Program section. Routines are the second element of the Program section.

Each Routine starts with the heading, “ROUTINE”, and ends with the heading, “END_ROUTINE”. Inside a routine, multiple rungs can be written. Each rung may have a Rung Description and Rung Code. Rung description starts with “RC” and Rung Code starts

with “N”, they both end with a semicolon (“;”). Every Program Section has a “Main” routine. This, “Main”, subroutine is used to call all other subroutines.

Figure 23 show an example of the Program section. The Program section contains Program Tag List and two Program subroutines. There is only one tag in the Program Tag List, and only one rung in both subroutines. Rung in the first routine has a Rung Description associated with it.

(NOTE: A complete Empty L5K file is shown in Appendix A.)

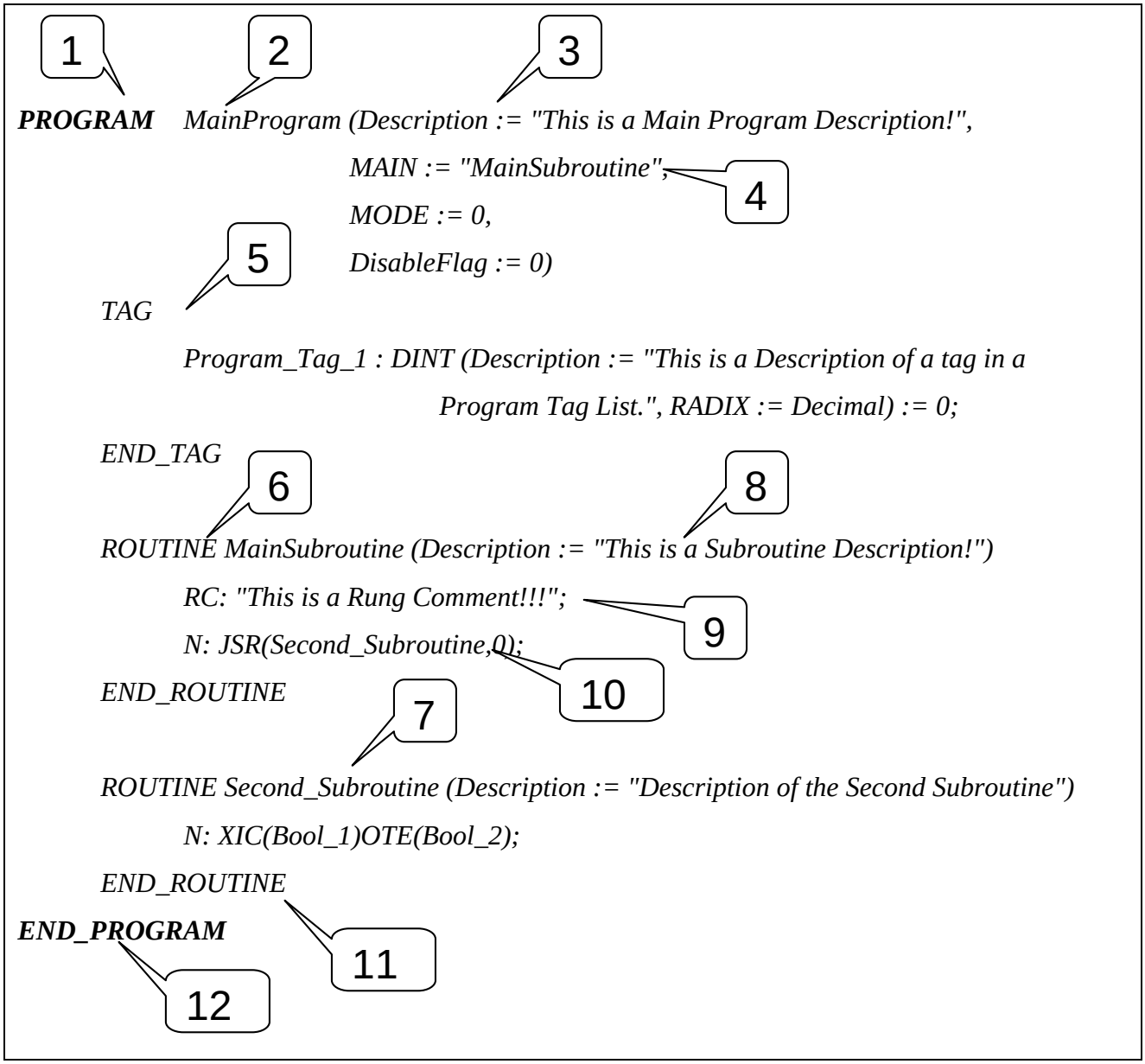


Figure 23 - L5K file Program Structure

1. Start of the Program section
2. Program Name
3. Description of the program
4. This is the main subroutine of the program. From this routine all other subroutines are called.
5. Program Tag List section
6. Start of the subroutine
7. Name of the subroutine
8. Description of the subroutine
9. Rung comment
10. Rung code (This particular code calls another subroutine. In this example, it calls subroutine called “Second_Subroutine”.)
11. End of a subroutine
12. End of a Program

3.2.3 Programming the L5K file

For the APCG software to be able to create a PLC code in the L5K file it is required to understand how the PLC instructions are written in the L5K file. There are hundreds of PLC instructions available [19], but not all of the PLC instructions are used in the projects. Only the PLC instructions that are commonly used by Realcold Milmech Pty. Ltd. engineers will be built-in in the APCG software.

An Instruction List (IL) language is not available in an RSLogix5000 program. The PLC languages available in the RSLogix5000 software are LD, SFC, FBD and ST. Representation of the instructions, from RSLogix5000 program, in the L5K file resemble the instructions in an Instruction List language. In Figure 24, instruction “MOVE” is shown. Value 10 is being moved into a tag called “Integer_1”.

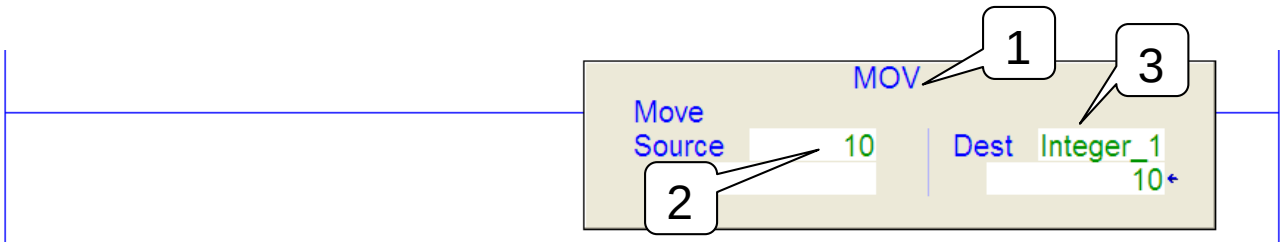


Figure 24 - PLC move instruction (LD language)

1. Instruction
2. Source (value 10)
3. Destination (“Integer_1”)

The L5K representation of the instruction shown in Figure 24, resembles IL language, as shown below.

- N: MOV(10,Integer_1);

(NOTE: Representation of most commonly used PLC instructions, by the engineers at Realcold Milmech Pty. Ltd., are listed in Appendix B.)

3.3 Visual Basic for Applications (VBA) programming

To create the APCG software, the Microsoft Excel (MS Excel) program will be used. The MS Excel has two equally important parts: Spreadsheets and VBA (Visual Basic for Applications). The MS Excel Spreadsheets have been used in the industry for many years and majority of people that use it are not “programmers”. VBA is a programming tool, which comes with MS Excel, and is mainly used by the programmers. VBA can be considered as a base of the MS Excel, which is not seen by most people. Macros are an automated version of the VBA programming, but it can only do simple things, and is not as powerful as directly programming in the VBA space. VBA is mainly used to automate MS Excel. It has its own “window”, Visual Basic Editor.

(Note: Explanation on how to use MS Excel Spreadsheets and VBA is outside the scope of this thesis.)

When creating the APCG software, both parts of the MS Excel will be used. MS Excel Spreadsheets will be used as a User Interface. The VBA part of the MS Excel will be used to program the functions of the APCG software.

3.3.1 User Interface

A User Interface will be used by an engineer to enter or select required information for the APCG software to create a PLC code.

A User Interface will have the Base (start up) sheets and the Engineered Sheets. The Base sheets are available in an empty, APCG software, which an engineer starts with at the beginning of programming. These sheets exist in every project, and the layout of the sheets does not change from project to project. There are nine sheets, which compose the Base sheets.

1. Title page – This sheet is used to enter a project name, number, description, and a date. Each sheet used in the APCG software program is also listed on this sheet, as a "table of content". The list is mainly used to quickly jump to a required sheet.
2. Digital Inputs sheet – All of the digital inputs used by the PLC program are to be entered on this sheet. In addition, each input has a function, such as an Indication only (i.e. Object present), an Alarm (i.e. Emergency Stop) or a Motor running.
3. Analogue Inputs sheet – Analogue inputs are entered here. If there are any alarms associated with an analogue input, than those need to be selected. The possible alarms are Low Cutout Alarm (LC), Low Alarm (LA), High Alarm (HA) and High Cutout Alarm (HC).
4. Digital Outputs sheet – Digital outputs and their function are to be entered on this sheet. Possible functions of a digital output are Motor (Motor Start), Solenoid (Solenoid Open) or another Actuator (i.e. Sounder, Light, et.). If motor is selected, "Motor Start", than the output needs to be linked with the digital input of the same motor.
5. Analogue Outputs sheet – Analogue outputs do not have a function associated with it.
6. I/O Tags List sheet – Once the PLC I/Os have been entered, an engineer will populate the I/O database by pressing a button.
7. PLC Internal Tags List sheet – While using, the APCG software, internal PLC tags will be required for programming, such as flags, timers, etc. The Internal PLC tags list will be used for such tags.

8. Flags sheet – This is a page for populating PLC flags subroutine. Flags subroutine can contain over a 100 flags, and this number of flags cannot fit on one sheet which means more than one sheet might be needed. If there is more than one sheet, in the APCG software program, than they are numbered Flags1, Flags2, etc. The Flags have been explained in detail in the previous section, 3.1.1.
9. SCADA/HMI sheet – This sheet is used for populating PLC SCADA/HMI subroutine. In this subroutine, tags are used by PLC and SCADA/HMI. More than one, of this, sheet can exist, just like Flags sheet(s) and for the same reason. The purpose of this subroutine has been explained, in detail, in section 3.1.1.

The Engineered sheets will not exist at start-up. Some of these sheets will be created automatically if associated with PLC I/O base sheets or the APCG software will automatically create them on an engineer's request. There are two types of sheets available for an engineer to create Control Sheets and Sequence sheets.

Control Sheets are used for creating the PLC I/O subroutines, as well as adding more Flags and SCADA/HMI sheets. In addition, any other subroutine, which does not have a sequence layout, should be done in a Control Sheet. The APCG software creates automatically, by pressing a button, the following sheets:

- Digital Alarms sheets – This is used to create a code for a Digital Alarms interlock. As it was explained, in section 3.1.1, this interlock is in series with the Input. The interlock incorporates any other variable that need to be taken into account before raising a Digital Input alarm.
- Analogue Alarms sheets – This sheet is used in the same manner as Digital Alarms Sheet(s). This is where all analogue alarm interlocks are programmed in.
- Motor sheets – On these sheets, Machine Interlocks and Start Permissive for motors, are programmed in. The Machine Interlocks and the Start Permissive for motors have been described at detail in section 3.1.1.
- Solenoid sheets – On these sheets, Machine Interlocks and Start Permissive for solenoids, are programmed in. The Machine Interlocks and the Start Permissive for solenoids have been described at detail in section 3.1.1.

- Sequence sheets – These sheets are used to create Sequence subroutines. Only one Sequence sheet is allowed for an actuator. The idea is to have moderate size Sequence subroutine, rather than a large and complicated one. The Sequence sheet is associated with Sequence controls, which allow an engineer to create Grafcet step design on the Sequence sheet.

For the APCG software to create a PLC code, an engineer will have to enter the required information in steps. The steps need to be followed, in the order described below, for the APCG software to be able to create a PLC code in an L5K file format.

1. Enter PLC I/Os and indicate the function of the PLC I/Os
2. Populate the PLC I/O Tags List sheet – the APCG software will populate the PLC I/O Tags List automatically when the engineer presses the correct button on the Control form.
3. Populate
   - Digital Alarms sheet(s) (Alarm Interlocks)
   - Analogue Alarms sheet(s) (Alarm Interlocks)
   - Motor sheets(s) (Machine Interlocks and Start Permissive)
   - Solenoid sheets(s) (Machine Interlocks and Start Permissive)
4. Create the Sequence sheets and design them in the Grafcet step format.
5. Populate the PLC Internal tags sheets as needed. This includes HMI/SCADA and Flags tags.
6. Populate the Flags and the SCADA/HMI sheets as needed
7. Create an L5K file – the APCG software will generate, the PLC code in an L5K file format, automatically when an engineer presses the correct button on the Control form.
8. Open the PLC code in the L5K file using RSLogix5000 software.
9. In the RSLogix5000 software change the controller type and add the PLC I/O cards in the program.
10. Link the PLC I/O cards with the appropriate PLC integers (created by the APCG software in step 2).

3.3.2 Visual Basic for Applications (VBA) programming

The data entered by an engineer in the APCG software needs to be converted into a L5K format, which a RSLogix5000 software can use. To create a L5K file with the date entered a Visual Basic for Application (VBA) software will be utilized. The code that will be written in the VBA needs to follow the steps an engineer will take, when creating a PLC code in a L5K file format using the APCG software.

Step 1:

- Does not use VBA

Step 2:

1. Go through all of the Digital Inputs. Check if the Digital Input is for an Indication, an Alarm, or a Motor. This is important, indicates where the input will be placed in the program. Structure of an input will be decided by this selection. In addition, the difference between the Indication and the Alarms is the Interlock bit on the Digital Input AOI instruction. If the input is an Indication, an Interlock bit for this input will not be placed on the Digital Input Control sheets.
2. Create tags for the Digital inputs, Digital input tag or a Motor structure tag. Because of the UDTs for Digital Inputs (that are not indication for motor running) and Motors, only one tag for each input type is required. UDT for Digital Inputs (that are not indication for motor running) incorporates inputs for indication as well as a digital input alarm.
3. Go through all of the Analogue Inputs and create the Analogue input tags. Only one tag is required for each analogue input. Analogue input UDT groups all relevant tags.
4. Go through the Digital Output tags. Check if the Digital Output is for a Solenoid, a Motor or another actuator. Create Digital Output tags for a Solenoid structure or link the output for a motor to the correct Motor structure.
5. Create integers for the PLC I/O cards (This is important in the Step 10).
    - DINT integer for each Digital card. All RSLogix5000 cards use DINT, regardless if the digital input card has 32 inputs or not. If the card has less than 32 inputs (i.e. 16 inputs), only first 16 bits of the DINT are used.
    - A REAL tag for each Analogue channel

Step 3:

The sheets mentioned below is created automatically when an engineer presses the relevant button on the Control form.

1. Create the Interlock sheet(s) for all the Digital alarms, if there are any.
2. Create the Interlock sheet(s) for all the Analogue Alarms, if there are any.
3. Create the Interlock and the Start Permissive sheet(s) for the Motors, if there are any.
4. Create the Interlock and the Start Permissive sheet(s) for the Solenoids, if there are any.

Step 4:

1. Create the Sequence Sheets, when requested by an engineer. An engineer will need to press a relevant button on the Control form, for the VBA to create this sheet.
2. Create Grafcet blocks and transition lines, when requested by an engineer. Buttons for each of the Grafcet blocks are on the Control form, for an engineer to use.

Step 5:

1. Populating internal PLC tags list, can be achieved in two ways:
    - Directly typing in all the requested information on the Internal PLC Tags List sheet
    - Typing the information on the Control form and pressing a button “Add a Tag”

Step 6:

- Does not use VBA

Step 7:

Create an L5K file is the most extensive part of the VBA code. This part combines all of the entered or created information and creates an L5K file. Two hidden sheets will be created for the VBA code to use, when creating L5K file.

First sheet, “L5K_Code”, contains all of the information that an empty L5K file has. This includes software version, processor information, and processor configuration; this has been explained in section 3.2.2. The information on this sheet does not change.

The second sheet, “L5K_Format”, is used for storing all the information, entered by an engineer, in L5K format. All of the tags and subroutines are stored in this sheet.

Both sheets are directly copied into an L5K file, which is created by a VBA code. One button on the Control form will be used to initiate an execution of this part of the VBA code. When the button is pressed the VBA code will execute following actions:

1. Create all of the tags, PLC I/O and internal PLC code tags, in the L5K format and store it in the "L5K_Format" sheet. This information is stored in the designated column of the sheet.
2. Create PLC subroutine for each Control Sheet(s) and store the information in the appropriate column of the "L5K_Format"sheet.
3. Create PLC subroutine for each Sequence Sheet and store the information in the appropriate column of the "L5K_Format"sheet.
4. Create a L5K file using Notepad.
5. Copy all of the information from the "L5K_Code" and "L5K_Format" sheets.
6. Save the L5K file in the location and under the name selected by an engineer.

Step 8:

- Does not use VBA

Step 9:

- Does not use VBA

Step 10:

- Does not use VBA

3.4 Testing of the Automatic PLC Code Generator (APCG) software

Once the APCG software is designed and created it needs to be tested. Two tests will be undertaken. In the first test, two things will be tested. The first part of the test will be to find if there are any "bugs" in the program, how easy is to use the software, does the software need more functions, does the designed layout have a nice flow and etc. The second part of the test will be to check the PLC code created by the APCG software. It is important that the L5K file can be used by the RSLogix5000 software, that the created PLC code has all the parts needed to function and that the PLC code follows the PLC standard of Realcold Milmech Pty. Ltd and the

standard of the research paper by Guttel et al [1]. A project selected for this test will be, a commissioned real life project, undertaken by Realcold Milmech Pty. Ltd.

For the second test an ongoing real life project, undertaken by Realcold Milmech Pty. Ltd. will be used. The Functional Description and PLC code will be created by using the APCG software. This test will be used for measuring the effectiveness of the APCG software. Time taken to crate and test a PLC code will be recorded and used to measure effectiveness of the APCG software. How effectiveness is measured is described in detail in section 3.4.2.

The design and commissioning of the projects, used for the testing, have been undertaken by the author of this thesis. Same project type, provided by Realcold Milmech Pty. Ltd. will be used in both tests. This will have two common factors when measuring the effectiveness of the APCG software.

3.4.1 Real Life Projects

The Automatic Carton Conditioning is one of the products that Realcold Milmech Pty. Ltd. sells. This product, called “Plate Freezer”, is used to freeze boxes of product (i.e. meet) within 24-hour period. Boxes are brought, on the conveyor, in front of the Plate Freezer. Those boxes are pushed into the plate freezer and the Plate Freezer will freeze the boxes over the 24-hour period. Once the boxes are frozen, the boxes are pushed out of the Plate Freezer onto the conveyor, which takes the boxes away.

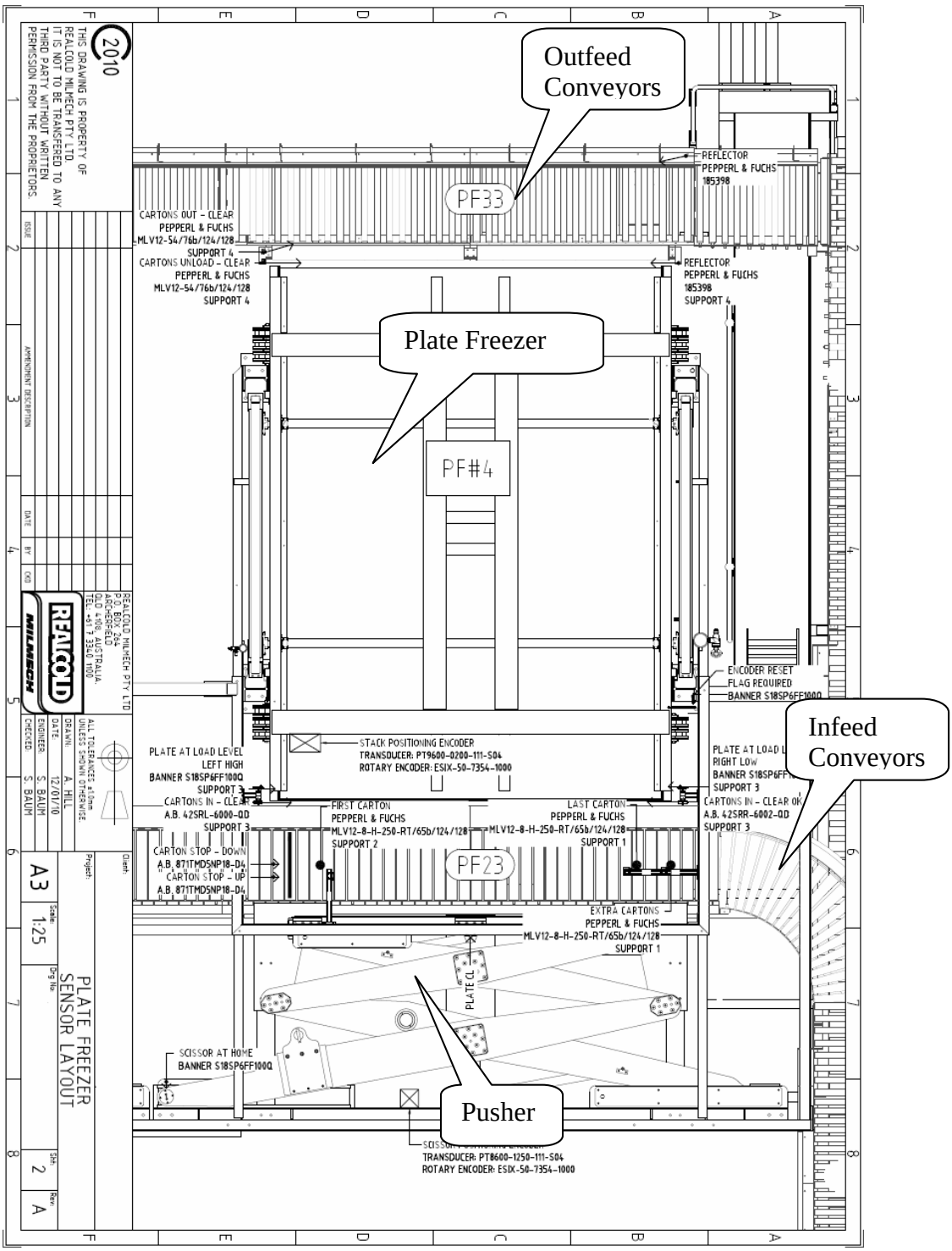


Figure 25 - Plate Freezer Layout Drawing

Figure 25 shows a layout of the Plate Freezer No.4. There can be number of Plate Freezer installed. Each Plate Freezer is the same as the one prior or after it. The layout shows how all the parts of the Plat Freezer are installed.

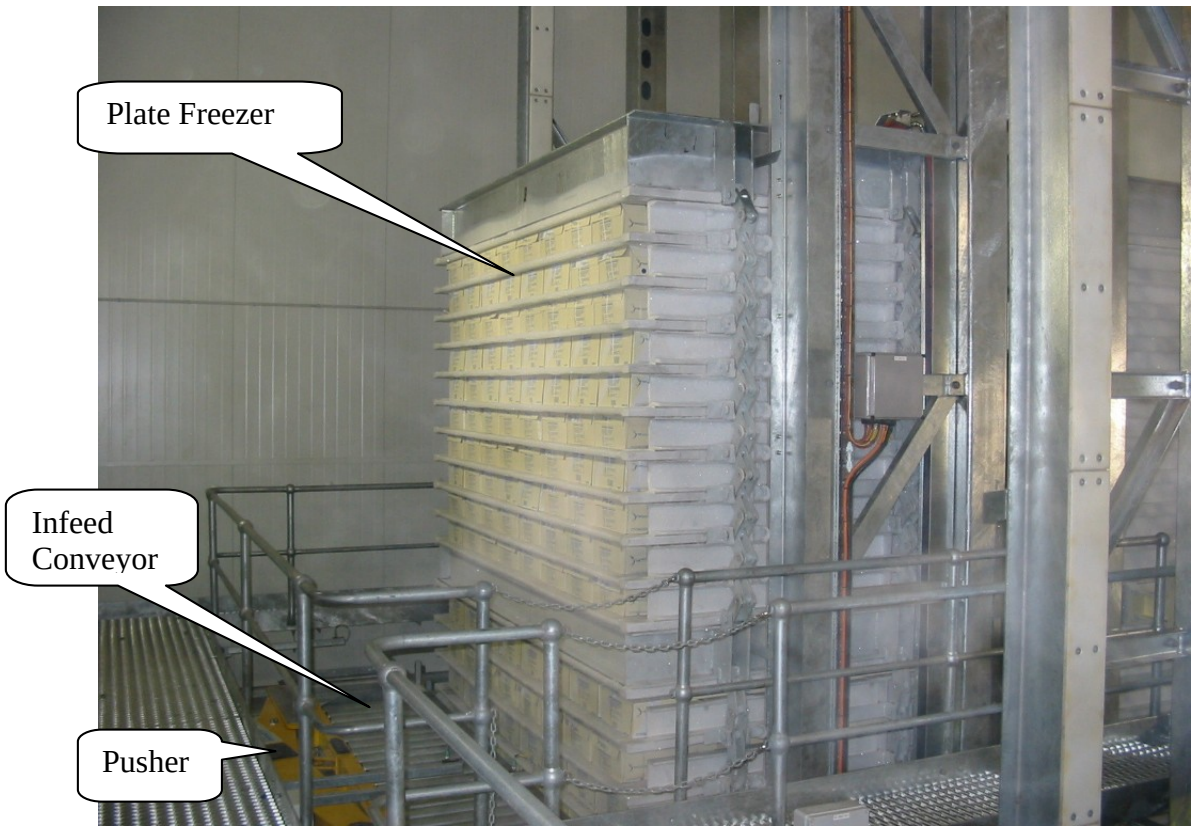


Figure 26 - Plate Freezer (Picture)

Figure 26 shows a Plate Freezer which is being loaded with the product. A pusher is retracting after loading one slug of cartons (8 cartons) into the Plate Freezer. The top half of the Plate Freezer is closed and the product in it is being frozen by the plates. The bottom half is open and is being loaded with the product.

These projects consist of conveyors, plate freezers and an LP Separator. An LP Separator is an Ammonia (NH3) vessel. Each Plate Freezer has one LP Separator associated with it. Ammonia liquid goes through each plate of the Plate Freezer. Normally there are at least three Plate Freezers installed with 18 levels (or 19 plates) for each Plate Freezer.

The project used in Test 2, has three Plate Freezers, LP Separator, In-feed conveyors and Out-feed conveyors. To control this machinery, over 150 PLC Inputs and Outputs are needed. This project will be divided into six areas, and each area will have multiple numbers of actuators associated with it.

1. Plate Freezer #1
    - 2 x Encoders

- 10 x Solenoids
- 3 x Motors
- 24 x Sensors (Digital)

2. Plate Freezer #2
    - 2 x Encoders
    - 10 x Solenoids
    - 3 x Motors
    - 24 x Sensors (Digital)
3. Plate Freezer #3
    - 2 x Encoders
    - 10 x Solenoids
    - 2 x Motors
    - 22 x Sensors (Digital)
4. In-feed Conveyors
    - 2 x Motors
    - 2 x Sensors (Digital)
5. Out-feed Conveyors
    - 2 x Motors
    - 2 x Sensors (Digital)
6. LP Separator Area
    - 2 x Motors
    - 1 x Solenoid (Digital)
    - 1 x Modulating Valve (Analogue)
    - 1 x Sensor (Analogue)
    - 6 x Sensors (Digital)

The PLC rack contains three High Speed Counter Cards, for encoders, three Digital Input cards, two Digital Output Cards, an Analogue Input card and an Analogue Output card.

- 3 x 1 High Speed Counter Cards - Encoders
- 3 x 32 Digital Input cards
- 2 x 32 Digital Output cards

- 1 x 8 Analogue Input cards
- 1 x 2 Analogue Output cards

PLC Inputs and Outputs List					
	Digital Sensors	Analogue Sensors	Motors	Solenoids	Actuators
Common to Plate Freezers	6	2	1	2	1
Plate Freezer #1	24	2	3	10	
Plate Freezer #2	24	2	3	10	
Plate Freezer #3	22	2	2	10	
In-feed Conveyors	4		2		
Out-feed Conveyor	4		2		
LP Separator	2	5	2	4	1
Total	**86**	**13**	**15**	**36**	**2**
Digital Inputs	**101**	Including Motors			
Analogue Inputs	**13**				
Digital Outputs	**51**	Including Motors and Solenoids			
Analogue Outputs	**1**				
Total I/O	**166**				

Figure 27 – Test 2 project PLC I/Os List

The Figure 27 shows a breakdown of the PLC I/Os needed for this project. There are 166 PLC I/Os needed, where 101 PLC I/Os are Digital Inputs, 51 PLC I/Os are Digital Outputs, and the rest are Analogue Inputs and Outputs. In addition, it can be seen that there are 15 motors and 36 solenoids in this project.

(NOTE: A complete PLC I/O list is shown in Appendix C)

3.4.2 Measuring the effectiveness of the APCG software

As mentioned earlier, "Time" will be used to measure effectiveness of the APCG software. The old cliché *"Time is Money"* will be used in measuring the cost savings while using the APCG

software. The less time is spent in designing, writing and testing of the PLC code, the more of the costs will be saved. To fully appreciate the time and cost savings that may occur when using the APCG software, it is necessary to understand the cost and time breakdown of a project. Four projects will be used to get a complete picture of the typical project breakdown concerning time and costs. Two projects will be based on Plate Freezers, while the other two will be different products that Realcold Milmech Pty. Ltd. provides. This is done to get a better understanding Engineering Costs have on a project. An average will be taken from four projects, to indicate the effect engineering costs have on a project.

- Project #1 – Plate Freezer Project
- Project #2 – Plate Freezer Project
- Project #3 – Refrigeration Project
- Project #4 – Refrigeration Project

The Figure 28 shows a breakdown in costs of any project undertaken by Realcold Milmech Pty. Ltd. It can be seen that any project can be broken down to four areas: Engineering Costs, Hardware Costs, Installation Costs and Warranty.

Engineering Costs are divided into nine areas:

1. Administration-Budgeting & Management – This area is used for Project Management. This includes time spent on getting the required information from Client, Contractors, Vendors, or other Engineers.
2. Engineering & Drawing Time – This is used to engineer an Electrical system. Once an Electrical system has been engineered, an Electrical Engineer will draw schematics for the system. These drawings are used by a MCC manufacturer and Electrical Installation contractors. These drawings are also given to a client.
3. Engineering Travel & Other Expenses – This is used if an Electrical Engineer needs to go to site during design stage.
4. Factory Acceptance Testing – Once the PLC and HMI/SCADA code are created than an Engineer needs to test it, before commissioning starts.
5. Functional Description – This is the first stage of the programming. To create a PLC and HMI/SCADA code, an understanding of the system is required. Functional Description describes the system in the Grafcet graphic format.

6. HMI Programming – SCADA & Touch – This is the time required to create a HMI/SCADA code and graphics.
7. Electrical Manuals – After a PLC and HMI/SCADA code have been designed, a manual is required for the operators and maintenance personnel, to be able to operate or fix the machinery.
8. PLC programming – This is the time required to write a PLC code.
9. Electrical Site Commissioning – This is the amount of time required by an Electrical Engineer to commission the machinery.

Hardware Costs are divided into five areas:

1. Field devices or other – Any sensors purchased by an Electrical engineer needs to be booked to this area ("bucket").
2. HMI Equipment – Purchase of HMI/SCADA runtime software and license as well as HMI/SCADA hardware is booked to this "bucket".
3. PLC Equipment – This is really the MCC equipment "bucket". All of the items that are located in the MCC are booked to this area, which includes PLC items, Power Supplies, safety relays etc.
4. VSD's & Softstarters – any VSDs or Softstarters purchased for a project is booked to this "bucket".
5. Subcontractors MCC & PFC – Purchase of MCC, Control Panels or Power Factor Correction is booked to this "bucket".

Installation Costs are divided into four areas:

1. Site Expenses – This is used for the costs an Electrical Engineer will create while commissioning. This includes accommodation, food, car petrol etc.
2. Site Travel Time – Some of the projects undertaken by Realcold Milmech Pty. Ltd. are commissioned overseas. Therefore, time required to get to commissioning site is booked to this "bucket".
3. Subcontractor Installation – Electrical installation and wiring is booked to this "bucket".

Warranty Expenses – Every installation has a minimum of 12 months warranty period. If any device breaks down it will be replaced free of charge to the client.

Project #1 - Plate Freezers					
Description	**Task Code**	**Budgeted Resource Hours**	**Budgeted Costs**	**Engineering Cost (%)**	**Total Costs (%)**
Administration-Budgeting & Mgt	ENG-ADM				
Engineering & Drawing Time	ENG-DWG				
Engineering Travel & Other Expenses	ENG-EXP				
Factory Acceptance Testing	ENG-FAT				
Functional Descriptions	ENG-FD				
HMI Programming -SCADA & Touch	ENG-HMI				
Electrical Manuals	ENG-MAN				
PLC Programming	ENG-PLC				
Electrical Site Commissioning	SIT-COM				
Total Engineering Costs					
Purchase-Field devices or othr	PUR-FLD				
Purchase-HMI Equip-includes PC	PUR-HMI				
Purchase-PLC Equipment	PUR-PLC				
Purchase-VSD's & Softstarters	PUR-VSD				
Subcontract M.C.C. & P.F.C.	SUB-MCC				
Total Hardware Costs					
Electrical site expenses	SIT-EXP				
Electrical site travel time	SIT-TRA				
Subcontract installation	SUB-INS				
Total Instalation Costs					
Electrical Warranty Expenses	WAR-ENG				
Total Costs					

Figure 28 - Empty Navision Table

(NOTE: Figures will be taken from the Realcold Milmech Pty. Ltd. Microsoft Business Solutions - Navision program. The tables have been modified to suite this thesis.)

There are three areas where the APCG software may save time and costs in the Engineering Section.

- Functional Description
- PLC Programming
- Factory Acceptance Testing

Ideally, it will be quicker to draw Functional Description, using tools available in the APCG software. In addition, the assumption is that the generated PLC code will be of a better quality, and will have fewer mistakes.

By using a Real Life Project as a test for the APCG software, it will be the best indication if the APCG software does save time and costs. By testing the APCG software in this fashion, the APCG software can be adjusted to better meet the needs of an engineer, as well as improve the functionality of the APCG software.

3.5 Summary

A PLC code can be divided into two parts: The Body and The Brain. The Body contains conditioning of the PLC Inputs and Outputs, Motors Structure, Solenoids Structure, Flags, HMI/SCADA interaction, and Communication. The Brain contains all of the Sequences of the PLC code.

PLC code can be viewed in two formats, using RSLogix5000 software, ACD file, or opening L5K file in Notepad. L5K file is normally not used by an Electrical Engineer for PLC programming. However, other software can use it, such as Visual Basic for Applications (VBA).

MS Excel and VBA will be used to create the APCG software. The APCG software will have two parts the User Interface and the VBA code. Both of these parts will be utilized to create a PLC code in the L5K format, which can be opened in RSLogix5000 software.

The designed and created APCG software will be tested in two different projects. In both tests a Real Life projects undertaken by Realcold Milmech Pty. Ltd. will be used. In the first test, the quality of the created L5K file will be checked. The second test will measure the effectiveness of the APCG software in a project. Time will be a measure of effectiveness.

4 Functionality of the APCG software

When the APCG software is opened for the first time, a Title page and Table of Contents are shown.

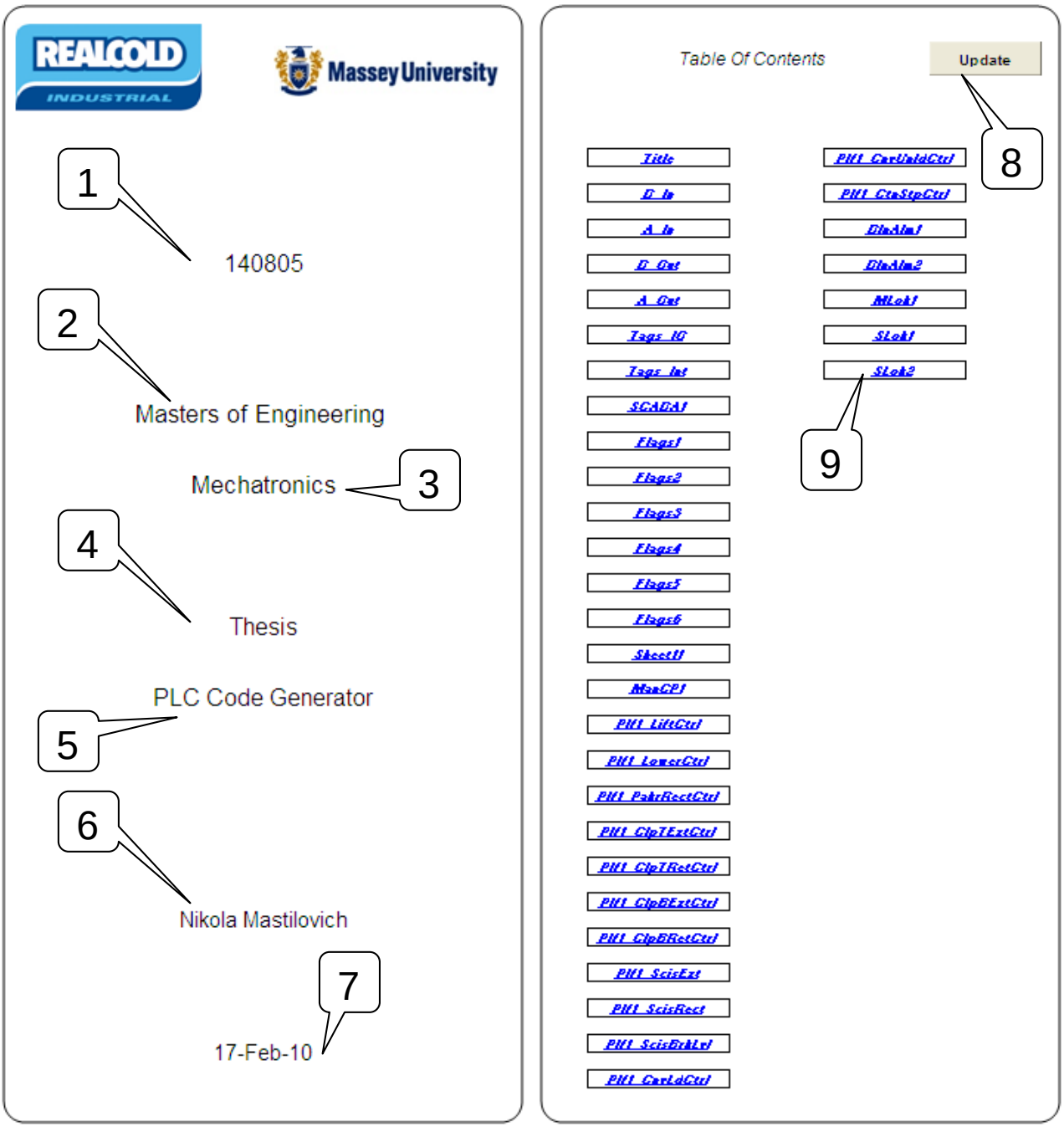


Figure 29 – APCG software Title Page

The information from the Title Page is used on every other page in the APCG software. Seven fields can be utilized for a project description, as shown in Figure 29.

1. Project Number
2. Project Name
3. Project Location
4. Project Description 1
5. Project Description 2
6. Project Engineer
7. Project Date

The “Table Of Contents” indicates how many sheets are used. This is also used as quick jump to any of the sheets.

8. Update – This button is used to update the Table of Contents
9. Click to jump to the page SLok2 (Solenoid Interlocks No.2 sheet)

4.1 Digital Input Screen

Figure 30 shows the user interface for the Digital Inputs. On this page, the engineer enters information for the Digital Inputs used in the project. In this section, each column will be explained.

There are four columns on the Digital Input screen and each one of them is used to indicate the inputs purpose:

1. Input Number
2. Symbol
3. Description
4. Function

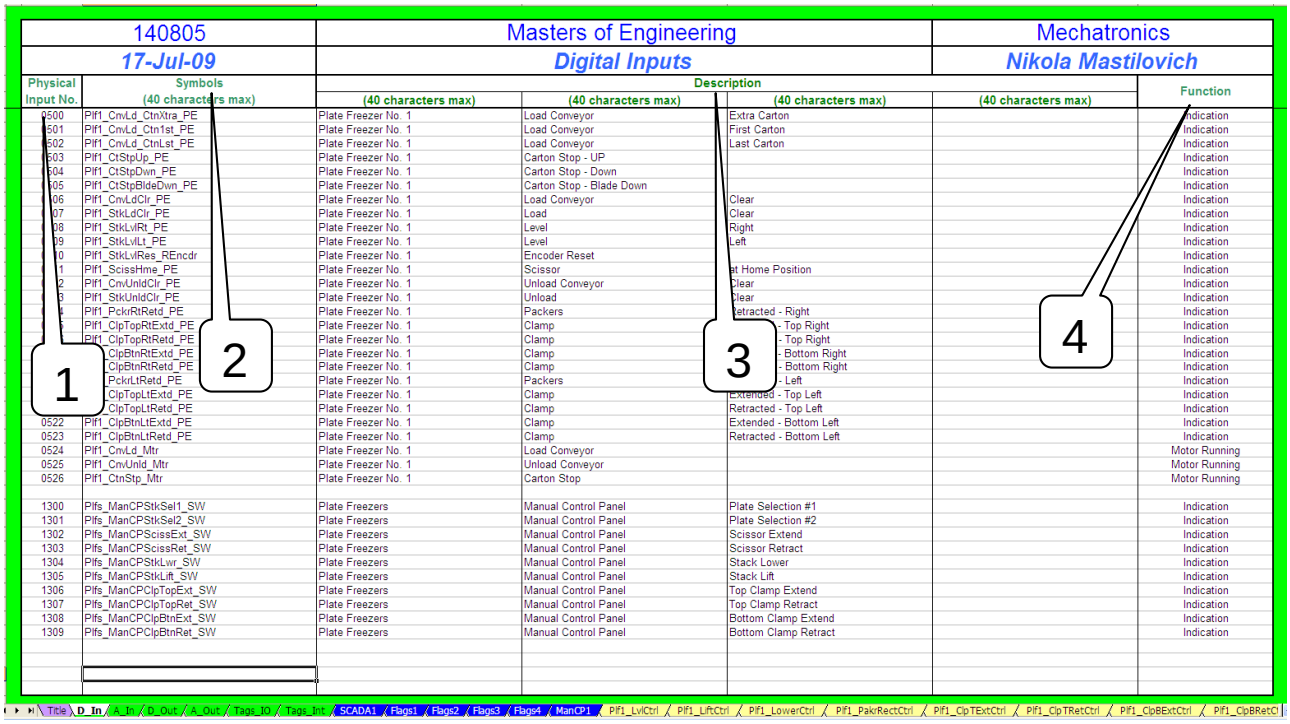


Figure 30 - Digital Inputs screen

1. **Input Number**

Input Number has two indications (Figure 31). First two numbers indicates the Slot number of the PLC card. The second two numbers indicates the Physical Input Number on the PLC card.

Example 0501

- 05 indicate that this digital input card is in slot 5 in the PLC rack
- 01 indicates 2nd input on the PLC card

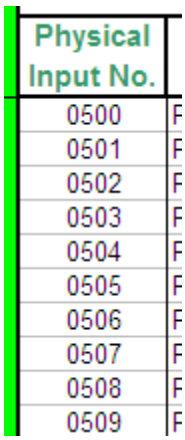


Figure 31 – Input Number

2. Symbol

Each tag, in the RSLogix5000 program, has its own symbol. Each symbol can have up to 40 characters (Figure 32).

Symbols (40 characters max)
Plf1_CnvLd_CtnXtra_PE
Plf1_CnvLd_Ctn1st_PE
Plf1_CnvLd_CtnLst_PE
Plf1_CtStpUp_PE
Plf1_CtStpDwn_PE
Plf1_CtStpBldeDwn_PE
Plf1_CnvLdClr_PE

Figure 32 - Tag symbol

3. Description

For each tag, there is a description. The APCG software allows up to four rows of description. Each row can contain 40 characters (Figure 33).

Description			
(40 characters max)	(40 characters max)	(40 characters max)	(40 characters max)
Plate Freezer No. 1	Load Conveyor	Extra Carton	
Plate Freezer No. 1	Load Conveyor	First Carton	
Plate Freezer No. 1	Load Conveyor	Last Carton	
Plate Freezer No. 1	Carton Stop - UP		
Plate Freezer No. 1	Carton Stop - Down		
Plate Freezer No. 1	Carton Stop - Blade Down		
Plate Freezer No. 1	Load Conveyor	Clear	

Figure 33 - Input Description

4. Function

Any input can be a safety input, an indication input, or an input for a motor. There is a variety of input functions, but the functions designed in the APCG software are the ones, which meet the standard of Realcold Milmech Pty. Ltd. (Figure 34).

Function
Indication
Alarm
Motor Running

Figure 34 - Input Function

4.2 Analogue Input Screen

140805		Masters of Engineering			Mechatronics	
17-Jul-09		*Analog Inputs*			*Nikola Mastilovich*	
Physical Input No.	Symbols (40 characters max)	Description (40 characters max)	(40 characters max)	(40 characters max)	(40 characters max)	Function
0100	Plf1_Stk_REncdr	Plate Freezer No. 1	Plate Encoder			LC LA HA HC
0101	Plf1_Sciss_REncdr	Plate Freezer No. 1	Scissor Encoder			LC LA HA

Figure 35 - Analogue Input Screen

Analogue Input screen does not greatly differ from the Digital Input screen.

Input Number section on the Analogue Input screen is the same as in the Digital Input section (Figure 36).

Physical Input No.
0100
0101

Figure 36 - Analogue Input - Input Number

The **Function** of the Analogue Input differs from the Function of the Digital Input. Under the Function column, a user selects if the Analogue input will be used to raise an alarm and if yes the user selects the type of the alarm (Figure 37).

Function
LC LA HA HC
LC LA HA

Figure 37 - Analogue Input - Function

The Analogue input can create four alarm types.

- Low Cutout Alarm – stops the machine
- Low Alarm – indicates that there is a problem
- High Alarm – indicates that there is a problem
- High Cutout Alarm – stops the machine

Not all of these alarms have to be used, so the user can choose which alarms are needed. Figure 38 shows the selection of alarms and possible combinations.

.

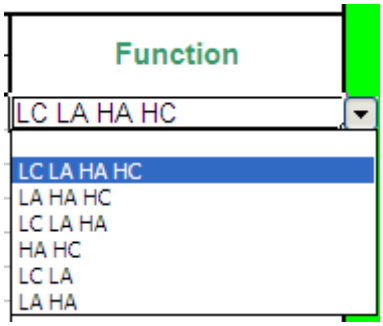


Figure 38 - Analogue Input - Function Options

4.3 Digital Output Screen

140805		Masters of Engineering			Mechatronics	
17-Jul-09		*Digital Outputs*			*Nikola Mastilovich*	
Physical Output No.	Symbols (40 characters max)	Description (40 characters max)	(40 characters max)	(40 characters max)	(40 characters max)	Function
0900	Plf1_ClpTopExt_Svo	Plate Freezer No1	Clamp Extend Control	(Top)		Actuator
0901	Plf1_ClpTopRet_Svo	Plate Freezer No1	Clamp Retract Control	(Top)		Solenoid
0902	Plf1_ClpBtnExt_Svo	Plate Freezer No1	Clamp Extend Control	(Bottom)		Motor Start
0903	Plf1_ClpBtnRet_Svo	Plate Freezer No1	Clamp Retract Control	(Bottom)		Solenoid
0905	Plf1_PckrRet_Svo	Plate Freezer No1	Peckers Retract Control			Solenoid
0906	Plf1_StkLift_Svo	Plate Freezer No1	Plate Lift Control			Solenoid
0907	Plf1_StkLwr_Svo	Plate Freezer No1	Plate Lower Control			Solenoid
0908	Plf1_ScissExt_Svo	Plate Freezer No1	Scissor Extend Control			Solenoid
0909	Plf1_ScissRet_Svo	Plate Freezer No1	Scissor Retract Control			Solenoid
0910	Plf1_CnvLd_Mtr	Plate Freezer No. 1	Load Conveyor			Motor Start
0911	Plf1_CnvUnld_Mtr	Plate Freezer No. 1	Unload Conveyor			Motor Start
0912	Plf1_CtnStp_Mtr	Plate Freezer No. 1	Carton Stop			Motor Start

Figure 39 - Digital Output Screen

The Digital Output screen does not differ from the Digital Input screen, shown in (Figure 39). As well as in the Digital Input screen, each output has its own function.

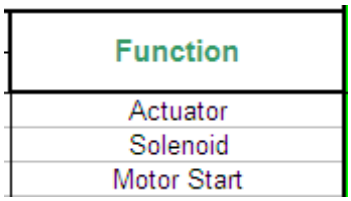


Figure 40 - Digital Output Function

Any Digital Output can be related to the Motor, Solenoid or another actuator. Figure 40 shows the possible selection for the Digital Output function.

4.4 Analogue Outputs Screen

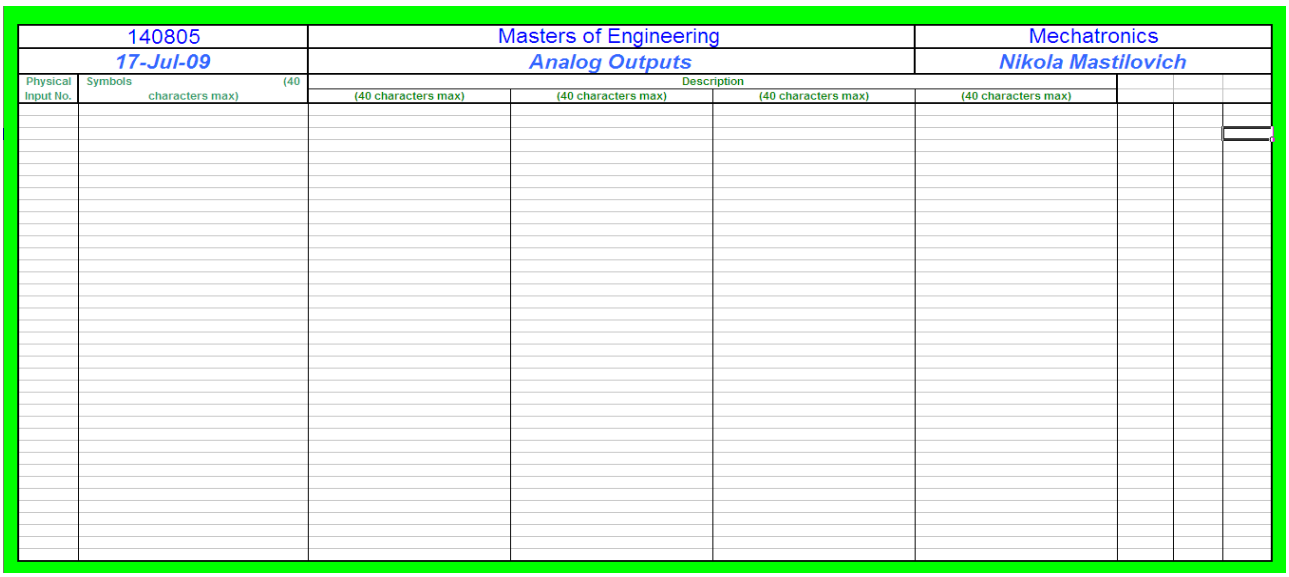
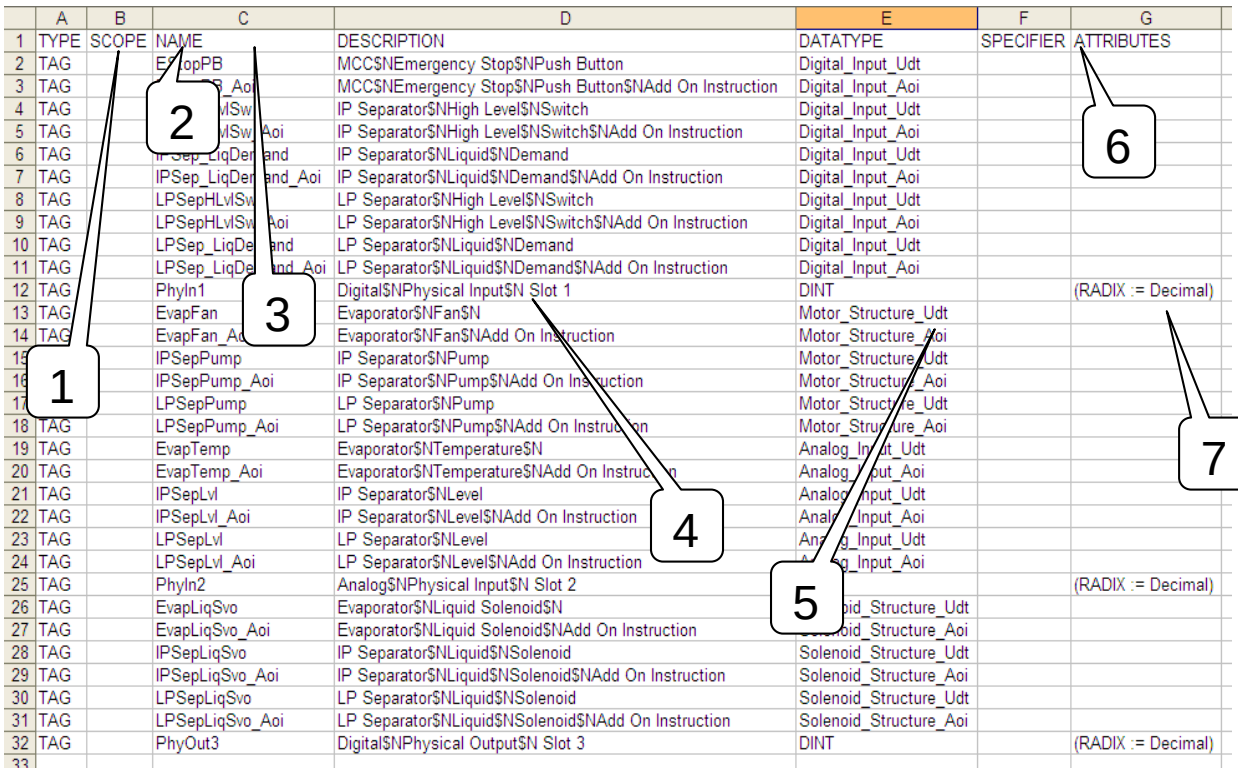


Figure 41 - Analogue Output Screen

Analogue Output Screen does not have a function column. The Analogue outputs are assumed, to be current outputs, 4-20mA (Figure 41).

4.5 PLC Internal and I/O Data Bases

The PLC I/O Database, that is populated by the APCG software, (Figure 42) looks like the database that the RSLogix5000 exports in the Microsoft Excel sheet [20]. This is not the L5K file database format. The APCG software will convert the database from the Microsoft Excel format into the L5K format when the L5K File button is pressed on Control Form, shown in section 4.7.



	A	B	C	D	E	F	G
1	TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE	SPECIFIER	ATTRIBUTES
2	TAG		EStopPB	MCC$NEmergency Stop$NPush Button	Digital_Input_Udt		
3	TAG		EStopPB_Aoi	MCC$NEmergency Stop$NPush Button$NAdd On Instruction	Digital_Input_Aoi		
4	TAG		IPSepHLvlSw	IP Separator$NHigh Level$NSwitch	Digital_Input_Udt		
5	TAG		IPSepHLvlSw_Aoi	IP Separator$NHigh Level$NSwitch$NAdd On Instruction	Digital_Input_Aoi		
6	TAG		IPSep_LiqDemand	IP Separator$NLiquid$NDemand	Digital_Input_Udt		
7	TAG		IPSep_LiqDemand_Aoi	IP Separator$NLiquid$NDemand$NAdd On Instruction	Digital_Input_Aoi		
8	TAG		LPSepHLvlSw	LP Separator$NHigh Level$NSwitch	Digital_Input_Udt		
9	TAG		LPSepHLvlSw_Aoi	LP Separator$NHigh Level$NSwitch$NAdd On Instruction	Digital_Input_Aoi		
10	TAG		LPSep_LiqDemand	LP Separator$NLiquid$NDemand	Digital_Input_Udt		
11	TAG		LPSep_LiqDemand_Aoi	LP Separator$NLiquid$NDemand$NAdd On Instruction	Digital_Input_Aoi		
12	TAG		PhyIn1	Digital$NPhysical Input$N Slot 1	DINT		(RADIX := Decimal)
13	TAG		EvapFan	Evaporator$NFan$N	Motor_Structure_Udt		
14	TAG		EvapFan_Aoi	Evaporator$NFan$NAdd On Instruction	Motor_Structure_Aoi		
15			IPSepPump	IP Separator$NPump	Motor_Structure_Udt		
16			IPSepPump_Aoi	IP Separator$NPump$NAdd On Instruction	Motor_Structure_Aoi		
17			LPSepPump	LP Separator$NPump	Motor_Structure_Udt		
18	TAG		LPSepPump_Aoi	LP Separator$NPump$NAdd On Instruction	Motor_Structure_Aoi		
19	TAG		EvapTemp	Evaporator$NTemperature$N	Analog_Input_Udt		
20	TAG		EvapTemp_Aoi	Evaporator$NTemperature$NAdd On Instruction	Analog_Input_Aoi		
21	TAG		IPSepLvl	IP Separator$NLevel	Analog_Input_Udt		
22	TAG		IPSepLvl_Aoi	IP Separator$NLevel$NAdd On Instruction	Analog_Input_Aoi		
23	TAG		LPSepLvl	LP Separator$NLevel	Analog_Input_Udt		
24	TAG		LPSepLvl_Aoi	LP Separator$NLevel$NAdd On Instruction	Analog_Input_Aoi		
25	TAG		PhyIn2	Analog$NPhysical Input$N Slot 2			(RADIX := Decimal)
26	TAG		EvapLiqSvo	Evaporator$NLiquid Solenoid$N	Solenoid_Structure_Udt		
27	TAG		EvapLiqSvo_Aoi	Evaporator$NLiquid Solenoid$NAdd On Instruction	Solenoid_Structure_Aoi		
28	TAG		IPSepLiqSvo	IP Separator$NLiquid$NSolenoid	Solenoid_Structure_Udt		
29	TAG		IPSepLiqSvo_Aoi	IP Separator$NLiquid$NSolenoid$NAdd On Instruction	Solenoid_Structure_Aoi		
30	TAG		LPSepLiqSvo	LP Separator$NLiquid$NSolenoid	Solenoid_Structure_Udt		
31	TAG		LPSepLiqSvo_Aoi	LP Separator$NLiquid$NSolenoid$NAdd On Instruction	Solenoid_Structure_Aoi		
32	TAG		PhyOut3	Digital$NPhysical Output$N Slot 3	DINT		(RADIX := Decimal)

Figure 42 - PLC I/O Data Base

1. TYPE – This field should always hold a string “TAG”, for each tag entered in the list.
2. SCOPE – This field should always be empty
3. NAME – The “name” is the symbol of a tag
4. DESCRIPTION – The APCG software allows four fields for the description.
5. DATATYPE – There is a great variety of data type used by RSLogix5000 software tags. Most commonly used tags, by Realcold Milmech Pty. Lt. engineers have been placed, as a selection, on the Control form.
6. SPECIFIER – This field should be empty
7. ATTRIBUTES – The Attributes column needs to be filled only if the Data type is not a UDT. “(RADIX = Decimal)” is the only item that should be entered in this column [20].

The Internal PLC Tags Database is populated by the engineer. This tag list consists of HMI/SCADA tags and internal tags required for the functioning of the PLC code, such as flags.

4.6 Flags and SCADA/HMI screens

The settings and controls of the Flags and the SCADA/HMI screens are the same. The Figure 43 shows Flags control screen.

1. Description on the page is the information of the Project. This information is automatically copied from the Title Page.
2. The Name of the screen, and thus this subroutine, is made out of two parts: Subroutine number and Subroutine Name.
3. Each subroutine has a description, and this is where description of the subroutine is entered.
4. There can be more than one screen for any Control Page. In the Figure 43, this is the first Flags control page. If there were more than one "Flags" control pages, all of the "Flags" control pages would be incorporated into one "Flags" subroutine.

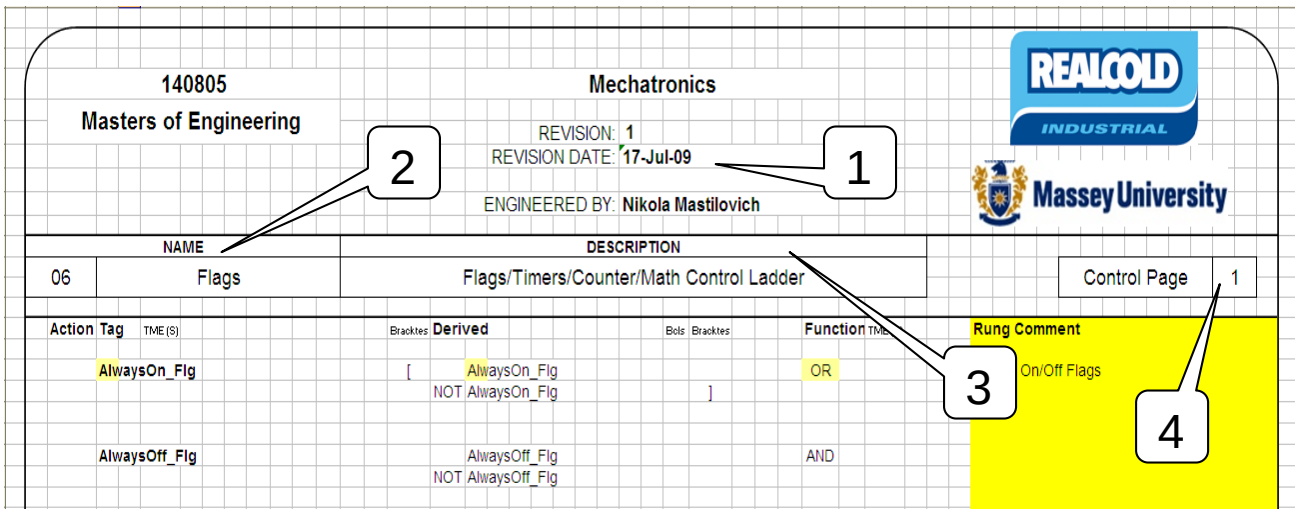


Figure 43 - Flags and SCADA/HMI screens

Flags and SCADA/HMI screens are part of the "base" screens. If more than one screen of each is required, an engineer will need to add more screens. More screens can be added by using Control form, which is explained in section 4.8.

The setup in the Figure 43 will create a subroutine *"_06_Flags"* in the PLC RSLogix5000, as shown in the Figure 44.

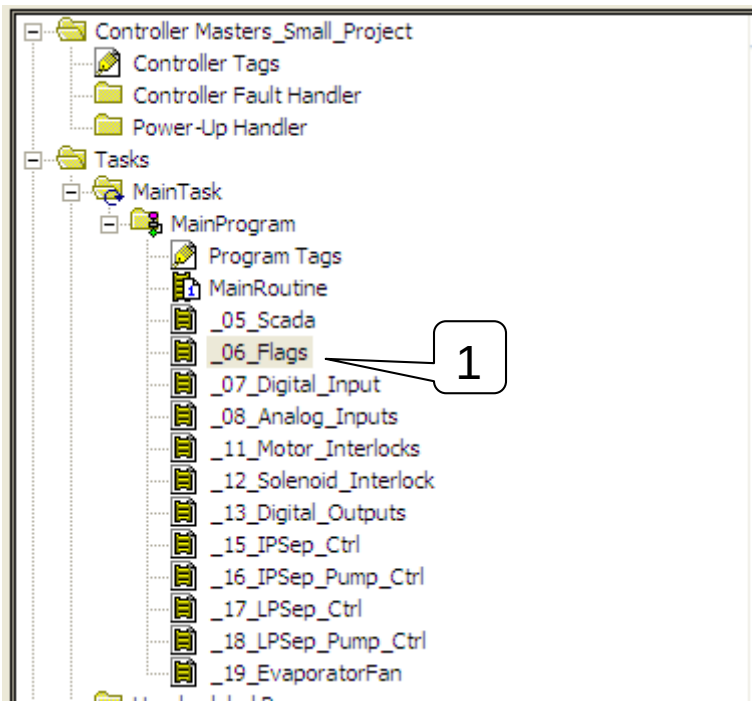


Figure 44 - ACD Flags

4.7 Control Pages

Control Pages are also used for the Interlocks and the Start Permissive. The Alarms, the Motors, and the Solenoids use the Control Pages (Figure 45).

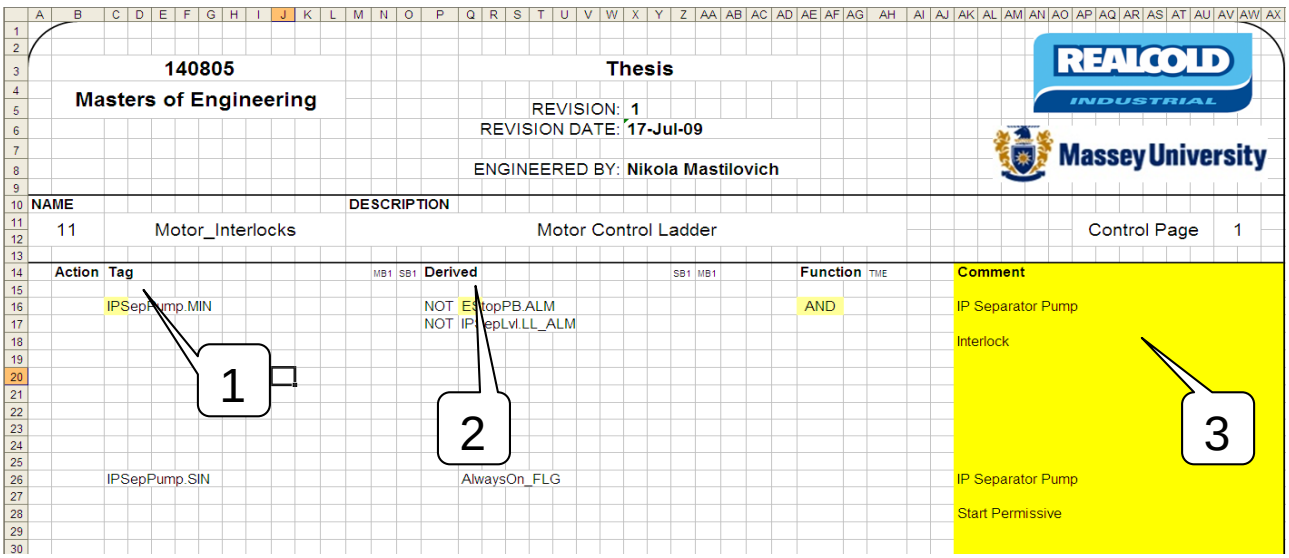


Figure 45 - Control Page

Control Pages for Machine Interlocks and Start Permissive are automatically created and initially populated by the APCG software, when an engineer pressed “Control Page” button on

the Control form (this is explained in the section 4.8). Once the pages are created, an engineer might require adding further conditions for Machine Interlocks or Start Permissive.

The Flags, the SCADA and the Interlock Control Pages are the same and have the same functionality.

1. **Tags and Action**

By looking at the Figure 45, Action and Tag columns are the first items that can be populated. This is the Output of the rung, which is the right most part of the rung (explained in section 3.2.3). Firstly, the user selects the Tag(s), which are to be controlled, and then how the tag(s) are to be used.

The Action column is where the user selects the instruction to be used, such as Add, Subtract, Multiply, Move, Timer On etc. Some of the instructions require one tag. Some instructions require two or three tags (Figure 46).

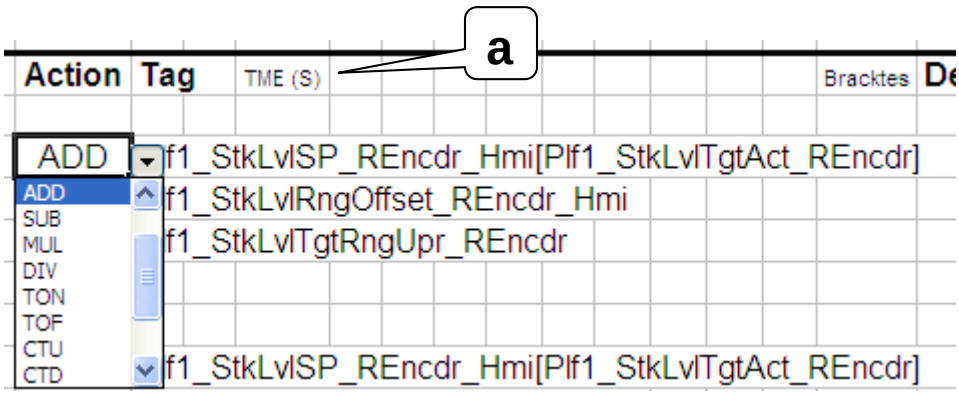


Figure 46 - Control Page - Action Column

a) The time column (TME(S)) is used with the Timers. This indicates the Preset value of the timer.

2. **Conditions**

The next section of the control page is to create conditions, as well as how these conditions will interact with one another (Figure 47).

Brackets	Derived			Bcls	Brackets		Function	TME (S)
[	(	Plf1_StkLvlTgtAct_REncdr					>	
		10		)			AND	
		Plf1_ClpTopRetd_PE_Flg					OR	
	(	Plf1_StkLvlTgtAct_REncdr					<=	
		10		)			AND	
		Plf1_ClpTopExtd_PE_Flg			]			

Figure 47 - Control Page – Conditions

The Brackets column is used when the conditions are parallel to one another, as was shown when alarms were described in section 3.1.1. As the interaction between conditions can be complicated and may require more than one square bracket, a selection of up to five square brackets can be used by the user, as shown in Figure 48.

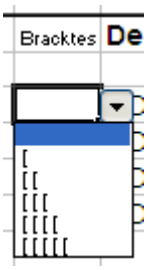


Figure 48 – Control Page –Square Brackets

The Derived column is linked with the Function. “NOT” and “ONS” are used to indicate Normally Closed instruction or One Shot Instruction (Figure 49). The “(“ is selected when compare functions are used (Figure 49).

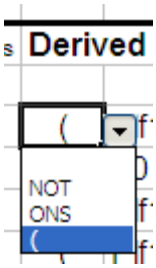


Figure 49 - Control Page - Derived

The Function column is used to select interaction between conditions, as well as to select comparison instruction (Figure 50). AFT and UNTL function is used for Delay ON or OFF timers. Sometimes a delay is required before the output action is to be executed. The TME(S) is

used when AFT or UNTL function is selected. This indicates the time that must pass before the action is executed (Figure 50).

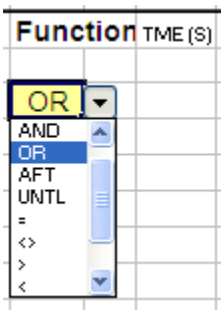


Figure 50 - Control Page - Function

3. Comment

As each rung can have a description associated with it, this is where the rung comment is to be entered (Figure 51).

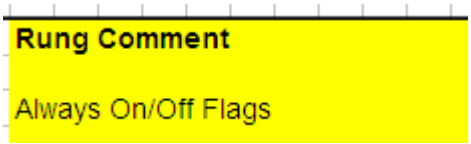


Figure 51 - Control Page - Rung Comment

4.8 Control Form

By pressing, "Ctrl+Z" keys on the keyboard, the Control Form appear (Figure 52). This form is used to create Control pages, Sequence pages, Sequence tools, and tags. There are five tabs on the Control Form.

Tab "Main"

1. *"PLC I/O Tags"* button is used to create all the I/O Tags from the physical inputs entered by the engineer in the Digital Inputs, Digital Outputs, and Analogue Inputs and Analogue Outputs screens.
2. *"Sort"* button is used to sort tags according to the Name or Data type. However, when PLC I/O Tags button is pressed the tags will be sorted by the sort selection "Name" or "Data type". "None" selection will not sort the tags but leave them as they appear on the Physical I/O pages.

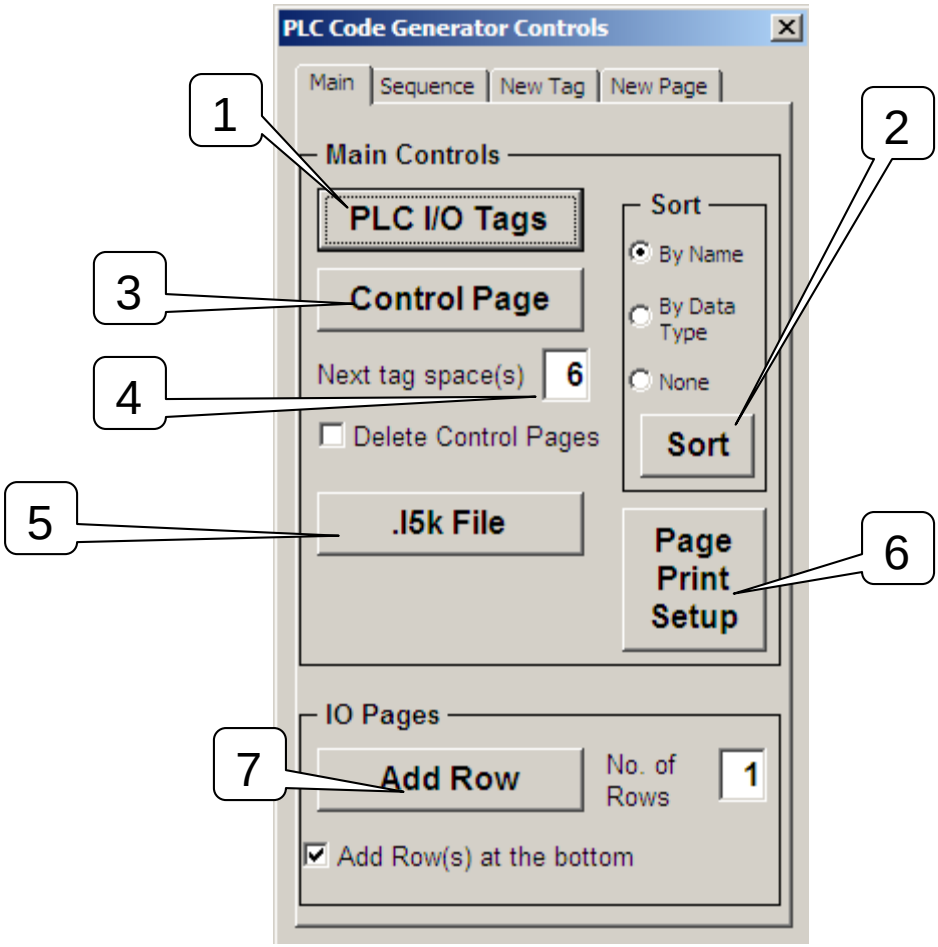


Figure 52 - Control Form – Tab Main

3. *"Control Page"* button is used when all of the Physical inputs have been entered. By pressing this button, all of the Control Pages will be created for alarms, motors, and solenoids (Figure 45).
4. As the Control Page is populated, the next Output tag will be created six cells below the last tag, leaving enough space for condition tags to be entered.
5. *"L5K file"* button is used to create the L5K file.
6. If any of the Control Pages or Sequence Pages do not print properly this button is used to adjust the print set up of the Active sheet.
7. When populating the I/O pages, it might be necessary to add more rows or add rows between entered symbols. The *"Add Rows"* button is used for that purpose. *"No. of rows"* textbox is used to enter how many rows will be added, while *"Add Row(s) at the bottom"* checkbox is used to select where the rows are to be added, at the bottom or below the selected row. When the Add Row button is pressed, it will only work if the

active sheet is one of the I/O sheets. If the active sheet is not an I/O sheet than the message pops up (Figure 53)

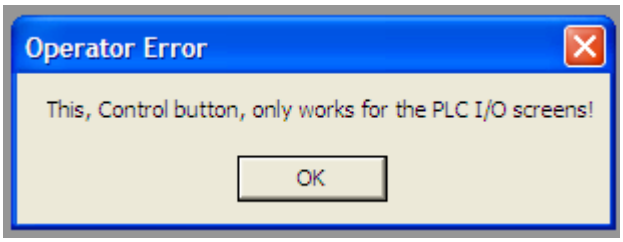


Figure 53 - Control Form-Add Row-Operator Error

Tab "New Tag"

1. In this textbox, the symbol is entered (Figure 55). There must be no spaces in a symbol name. If the space exists in the name, a massage will pop up (Figure 54). If the tag already exists in the list the user will be notified (Figure 54).

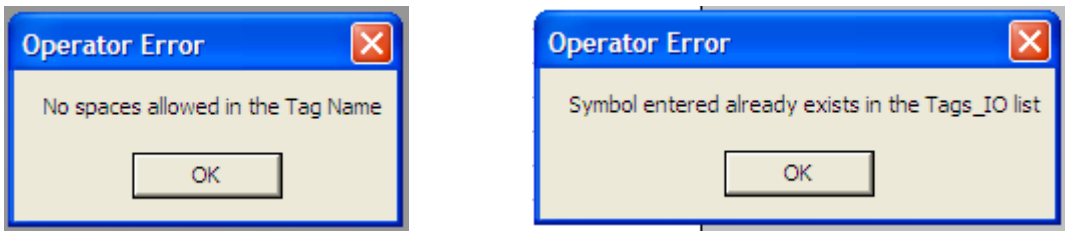


Figure 54 - Control Form-New Tag-Error

2. For each tag, a data type needs to be selected. The most common data types used by the Realcold Milmech Pty. Ltd. engineers have been put as a selection on the Control Form.
3. For all other data types or User Defined Data Types (UDTs), a UDT needs to be selected and the name of the data type entered.
4. If an array is needed than an array checkbox needs to be selected and a value entered for the size of the array.
5. Four textboxes are allocated for the description
6. Once all of the information has been entered, the *"Add Tag"* button will add a tag to the bottom of the Internal PLC Tag list.

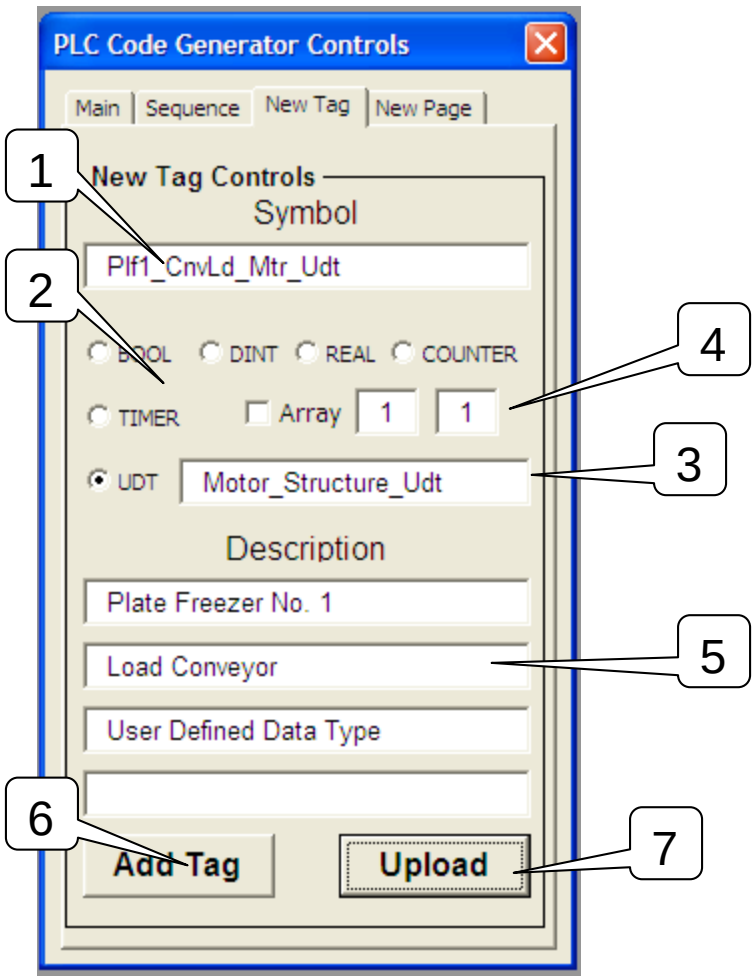


Figure 55 - Control Form-New Tag

7. *"Upload"* button is used to upload tag information. A particular tag needs to be selected on the active sheet, and the tag information will be uploaded. If the tag does not exist or the tag was not selected the user will be notified (Figure 56).

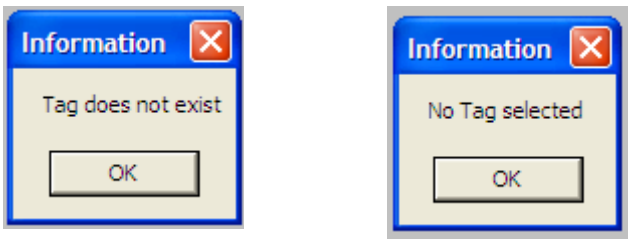


Figure 56 - Control Form-New Tag-Notification

Tab "New Page"

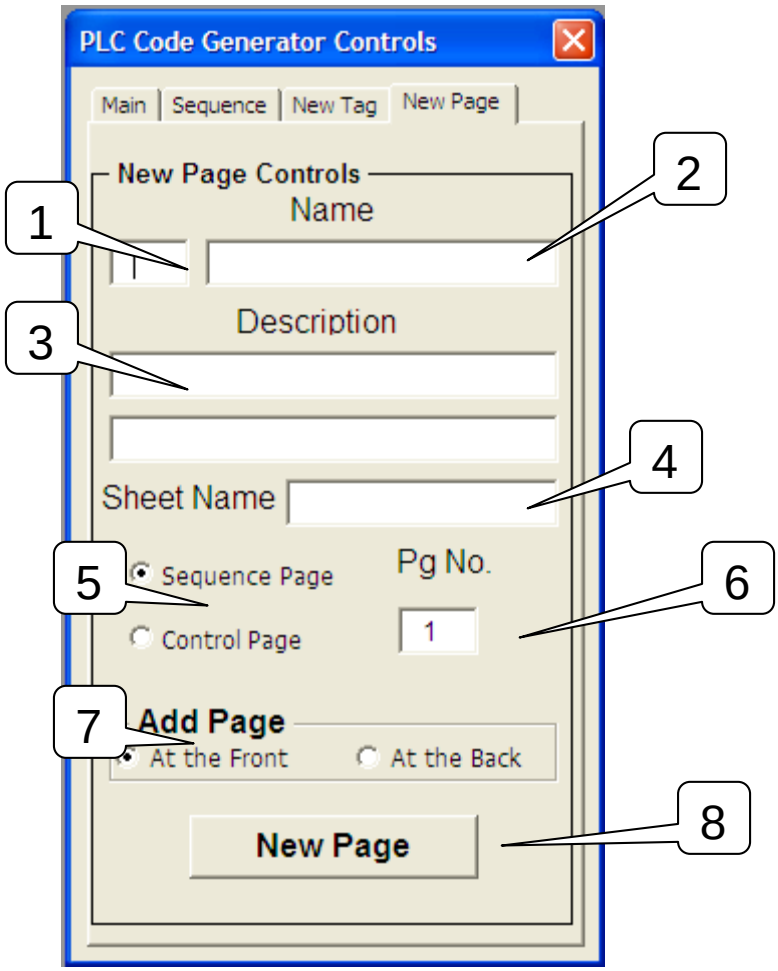


Figure 57 - Control Form - New Page

1. Each PLC subroutine starts with the number followed by the name (Figure 43 and Figure 44)
2. This is the Name of the PLC subroutine. The Name cannot contain any special characters, such as #, $, &, etc.
3. Description of the PLC subroutine
4. Each sheet in the MS Excel has a name. When a new sheet is created, MS Excel's generic name is Sheet1, Sheet2, etc. These generic names are not very useful, so entered *"Sheet Name"* will replace the generic name.
5. Selected a new page's type, Sequence or Control page

6. When populating the Control Pages, sometimes the information cannot fit on the one sheet. When more than one sheet is required, the page number indicates what page in the sequence the sheet is.
7. This indicates the location of the new sheet, at the front or at the back. Each sheet can be manually moved to the wanted location.
8. By pressing the *"New Page"* button, a new page is created.

If the user does not enter a Page Number, Page Name or Sheet Name the user will be notified to do so (Figure 58)

Figure 58 - Control Form - New Page-Notification

Tab "Sequence"

Information on the Sequence tab will be explained in the next section (Sequence pages), since these controls are used on the sequence page.

4.9 Sequence Pages

The sequence page is drawn in the Grafcet layout. This means that the program goes from one step to the next, and each step might control an output.

1. When the *"Add Step"* button is pressed, a step is added to the Sequence page. Prior to pressing the button column *"J"* needs to be selected. If the button is creating a first step, than the cell selected needs to be on row 17.

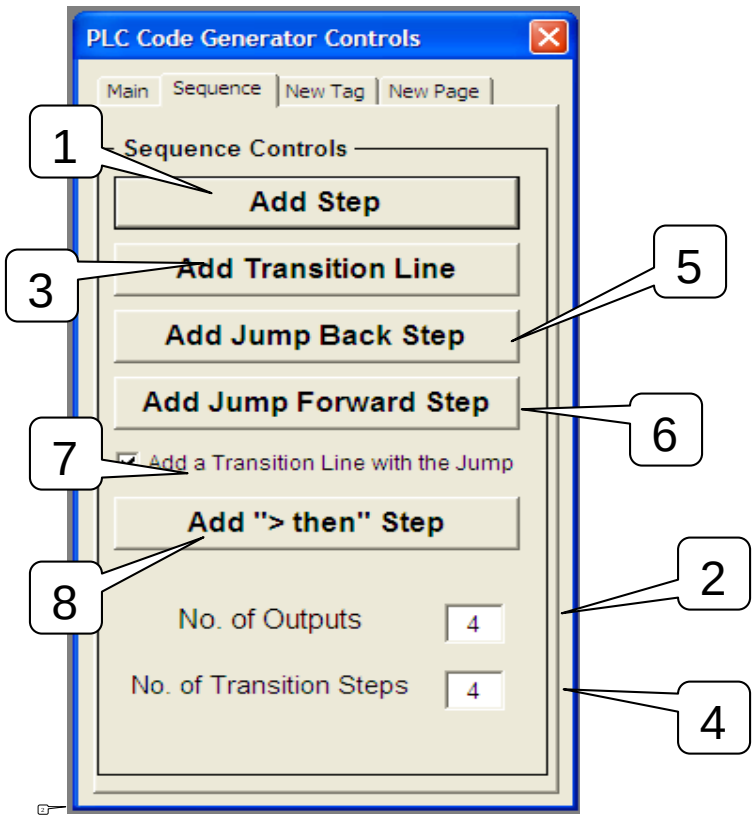


Figure 59 - Control Form – Sequence

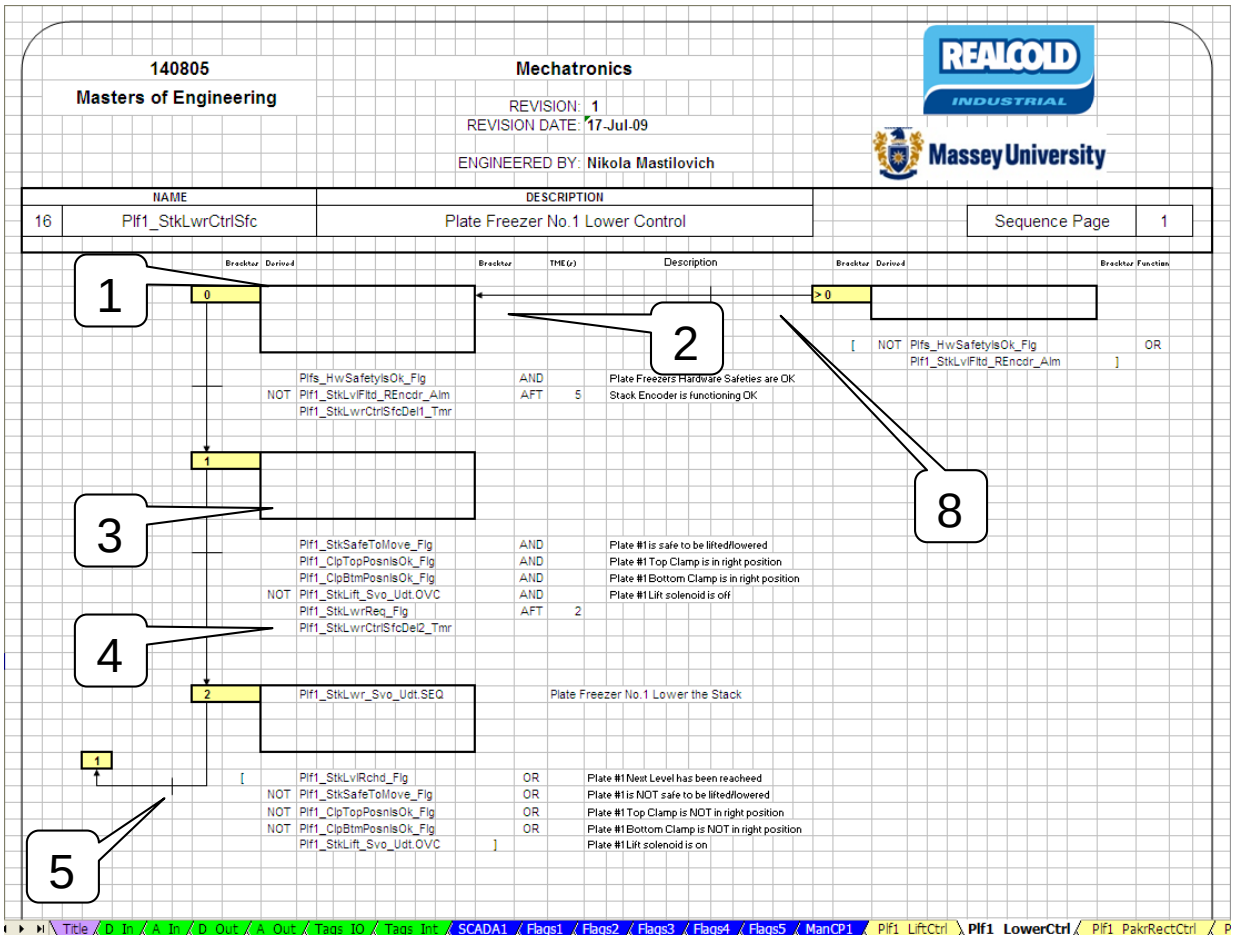


Figure 60 - Sequence Page

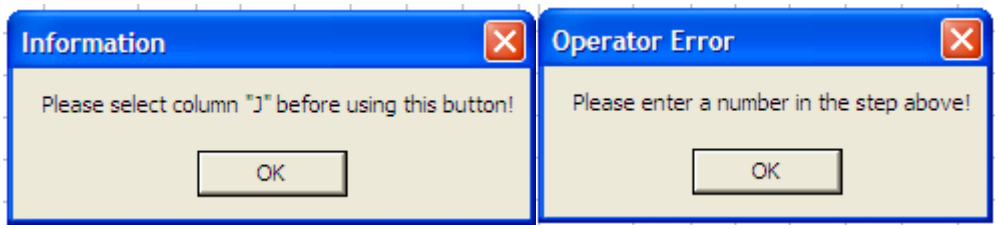


Figure 61 - Sequence Page-Notification

If a column, other than *"J"* is selected, a message will pop up when the *"Add Step"* button is pressed. When the first step is to be created, the user selects a row, other than 17, the message will pop up suggesting that the step above does not have a step number, which it does not since there is no step (Figure 61). This message is also an indication that a step number is missing. Initially when the step is created the APCG software will allocate a step number, starting from step 0, or a next number will be selected based on the previous step.

2. A step can control zero rung outputs or a number of rung outputs. Therefore, a step may need more or less than four rows (Figure 62). The textbox in Figure 59 is used to select the number of rows needed by the step.

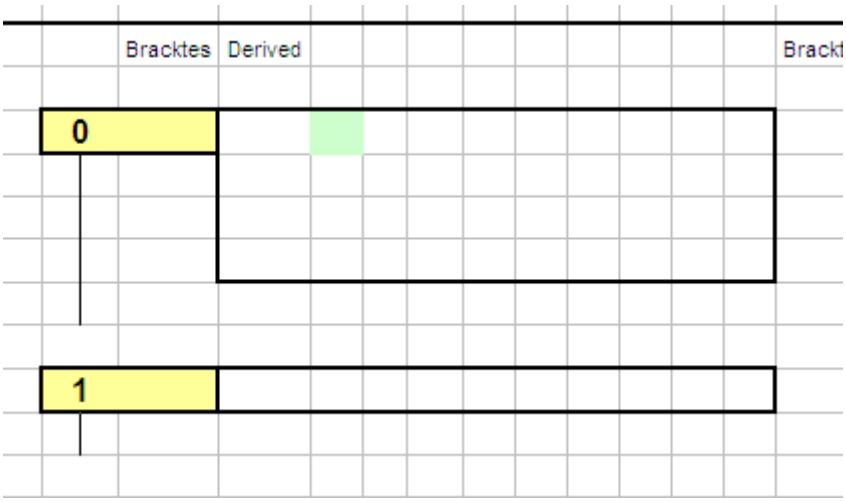


Figure 62 - Sequence Page-Step Size

3. A transition line is used alongside the rung conditions. This line has an arrow at the bottom of it.'
4. A transition line might also need to be bigger or smaller in length. The textbox *"No. of Transition Steps"* (Figure 59) is used to enter a length of the transition steps. In Figure 60, two different transition lines, which have different line lengths, can be seen.

5. Sometimes it is required to jump to the step before the current step. The *"Add Jump Back Step"* is used to create a jump back step (Figure 60).
6. The *"Add Jump Forward Step"* does the same thing as *"Add Jump Back Step"* button but in the reverse direction, that is, the forward direction (Figure 63).

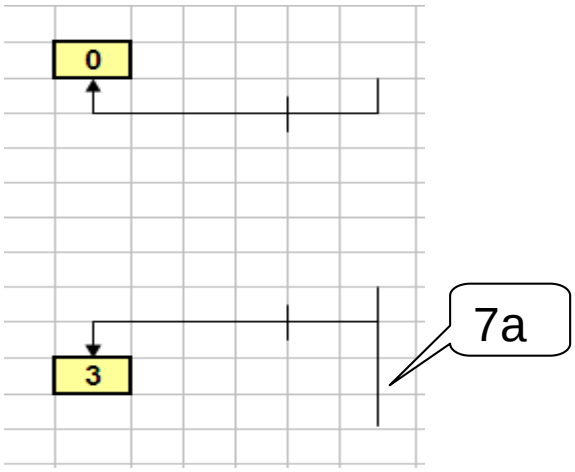


Figure 63 - Sequence Page - Jump Steps

7. If the jump is not the last part of the sequence than a transition line needs to be created with the jump step (7a).
8. Part of the safety feature of the sequence is to be able to jump to first step (step 0) from any step (i.e. from *"Greater than"* 0 step) (Figure 60). This usually occurs when a safety feature is triggered, such as Emergency Stop. To insert a *"Greater than"* step, the user needs to select a step where the program will return to when *"Greater than"* conditions have been met. This step can be used on any step, but it is most commonly used for "step 0".

4.10 Functionality of the Sequence Pages

The functionality of the Sequence Page does not greatly differ from the functionality of the control page. The main difference is the layout of the page itself.

Regarding the conditions of in-between steps, it has the same functionality as conditions on the control page. As in the control page, there is a Bracket, a Derived and a Function column (Figure 64).

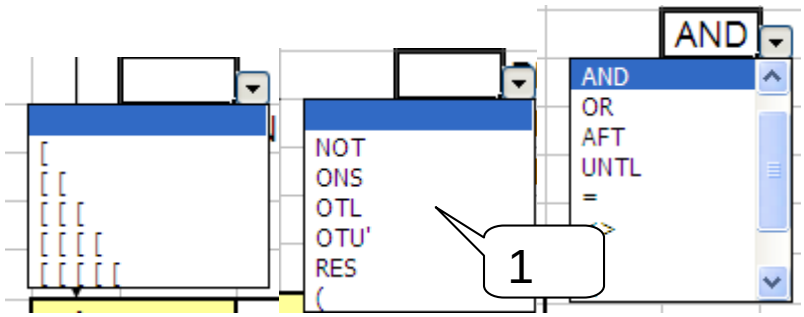


Figure 64 - Sequence Page - Conditions

1. The derived column, in the Sequence page, has extra instructions than in the Control page. These new instructions are used in the sequence steps, and not for the conditions. As each step can control a rung output, OTL (Output Latch), OTU (Output Unlatch) and RES (Reset Timers/Counters) are used for the control of the output.

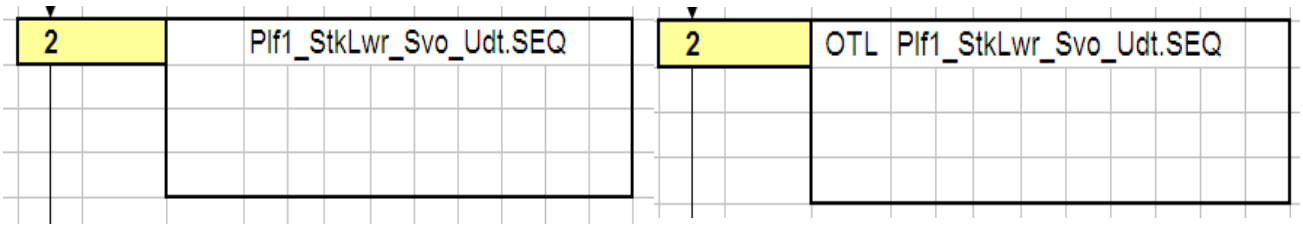


Figure 65 - Sequence Step - Step Output

If there is no instruction prior to the controlled rung output, than that output is ON while the step is active, and OFF when the step is not active.

To be able to understand the Sequence, an area has been allocated for the description. This is not used by the APCG software. It is there for the user to enter description, which can explain in more detail what the sequence does. This is shown in Figure 66 by pointer “a”.

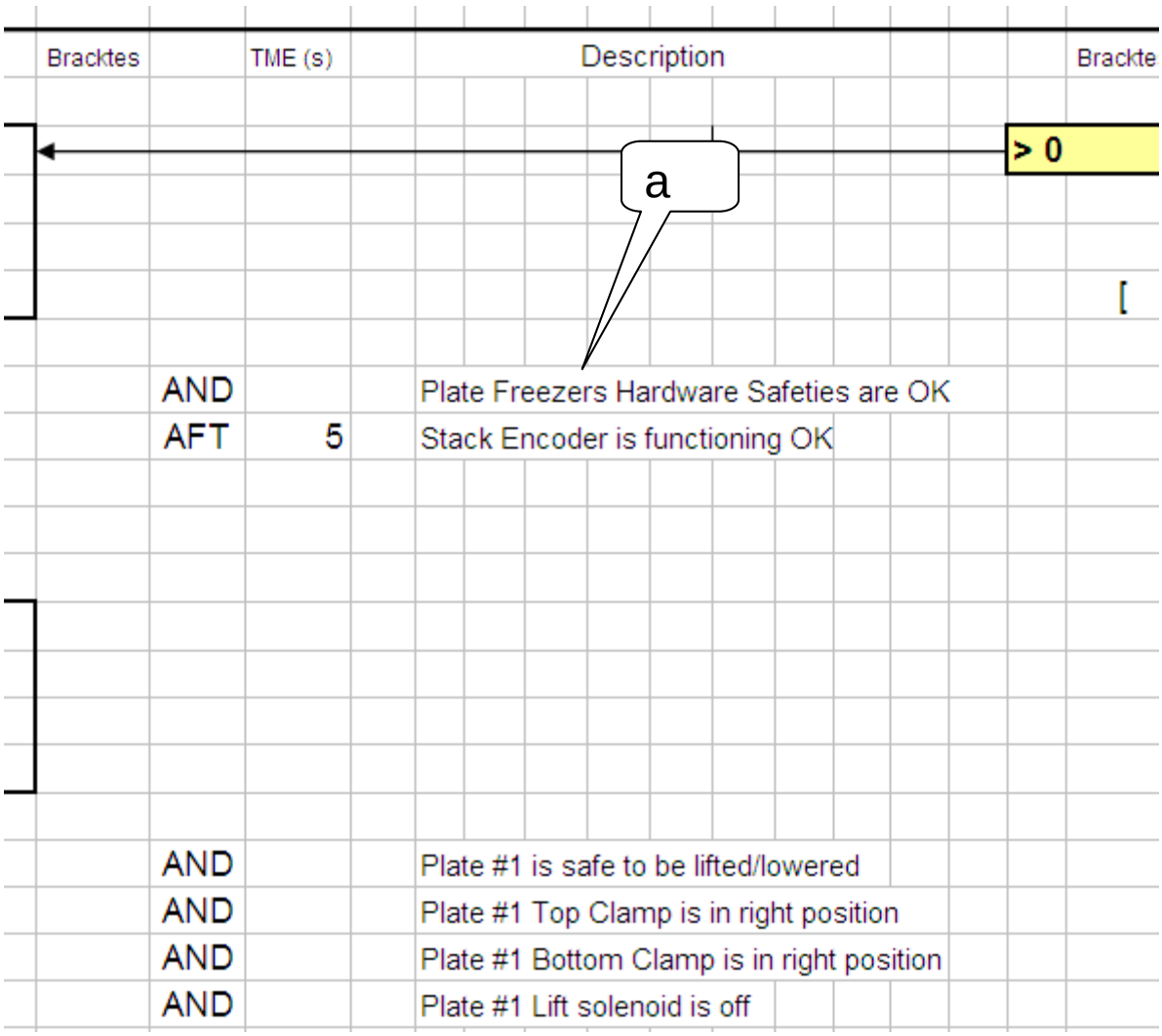


Figure 66 - Sequence Page – Description

4.11 Summary

In this chapter the APCG software was explained. All of the functions and steps needed to create a PLC code with the APCG software were detailed.

The APCG software has 9 base screens. An engineer uses these screens to enter PLC I/O information. Using a Control form an engineer is enabled to create Alarms, Motors and Solenoids Interlock screens. Sequence screens are also created by the use of Control form.

Once all the relevant data has been entered the APCG software creates L5K file which is then opened in the RSLogix5000 software for further PLC code development.

5 The APCG software – Real Life Projects

To understand the effectiveness of using APCG software it is required to understand what areas the APCG software can influence and how important those areas are. Time was used as a measure of effectiveness.

It has been hypothesised that the APCG software can influence three areas in the Engineering part of the project. Those areas are:

1. Functional Description (FD) – Functional Description is a sequence written in Grafcet graphic format. This is the first step when writing a PLC code.
2. PLC programming
3. Factory Acceptance Testing (FAT) – Once the PLC code has been written, it needs to be tested, before commissioning starts.

Time needed to create a PLC code with the APCG software was recorded, and compared to the budgeted costs of the relevant areas. By converting Time saved to money, by using APCG software, a true effectiveness of the APCG software was measured. At present (date when the budget was taken), the client was charged NZ$80.00 per hour.

1 hour = NZ$80.00

5.1 Project Budgeted Costs

To get an understanding of the costs involved in projects, four projects, of two different product type were used. All four projects have been undertaken by Realcold Milmech Pty. Ltd. Budgeted costs, of the four projects used, were averaged. This gave clearer picture of the costs involved with a project.

- Project #1 – Plate Freezer Project (shown in Figure 67)
- Project #2 – Plate Freezer Project (shown in Figure 68)
- Project #3 – Refrigeration Project (shown in Figure 69)
- Project #4 – Refrigeration Project (shown in Figure 70)

Project #1 - Plate Freezers					
Description	**Task Code**	**Budgeted Resource Hours**	**Budgeted Costs**	**Engineering Cost (%)**	**Total Costs (%)**
Administration-Budgeting & Mgt	ENG-ADM	66	$5,280.00	7.42	
Engineering & Drawing Time	ENG-DWG	107	$8,560.00	12.03	
Engineering Travel & Other Expenses	ENG-EXP		$6,198.24	8.71	
Factory Acceptance Testing	ENG-FAT	**33**	**$2,640.00**	3.71	0.86
Functional Descriptions	ENG-FD	**50**	**$4,000.00**	5.62	1.31
HMI Programming -SCADA & Touch	ENG-HMI	133	$10,640.00	14.95	
Electrical Manuals	ENG-MAN	30	$2,400.00	3.37	
PLC Programming	ENG-PLC	**159**	**$12,720.00**	17.88	4.16
Electrical Site Commissioning	SIT-COM	234	$18,720.00	26.31	
Total Engineering Costs		**812**	**$71,158.24**		**23.27**
Purchase-Field devices or othr	PUR-FLD		$3,423.53		
Purchase-HMI Equip-includes PC	PUR-HMI		$5,216.56		
Purchase-PLC Equipment	PUR-PLC		$18,461.14		
Purchase-VSD's & Softstarters	PUR-VSD		$0.00		
Subcontract M.C.C. & P.F.C.	SUB-MCC		$31,260.88		
Total Hardware Costs			**$58,362.11**		**19.09**
Electrical site expenses	SIT-EXP		$6,494.24		
Electrical site travel time	SIT-TRA		$0.00		
Subcontract Installation	SUB-INS		$166,894.54		
Total Instalation Costs			**$173,388.78**		**56.71**
Electrical Warranty Expenses	WAR-ENG		**$2,843.37**		**0.93**
Total Costs			**$305,752.50**		

Figure 67 - Project #1 Budget

Figure 67 shows the Budget allocated for first Plate Freezer project.

Project #2 - Plate Freezers					
Description	**Task Code**	**Budgeted Resource Hours**	**Budgeted Costs**	**Engineering Cost (%)**	**Total Costs (%)**
Administration-Budgeting & Mgt	ENG-ADM	112.5	$9,000.00	9.42	
Engineering & Drawing Time	ENG-DWG	150	$12,000.00	12.57	
Engineering Travel & Other Expenses	ENG-EXP		$34,820.00	36.46	
Factory Acceptance Testing	ENG-FAT	40	$3,200.00	3.35	0.94
Functional Descriptions	ENG-FD	23.75	$1,900.00	1.99	0.56
HMI Programming -SCADA & Touch	ENG-HMI	50	$4,000.00	4.19	
Electrical Manuals	ENG-MAN	26.25	$2,100.00	2.20	
PLC Programming	ENG-PLC	75	$6,000.00	6.28	1.76
Electrical Site Commissioning	SIT-COM	281	$22,480.00	23.54	
Total Engineering Costs		**758.5**	**$95,500.00**		**27.96**
Purchase-Field devices or othr	PUR-FLD		$8,000.00		
Purchase-HMI Equip-includes PC	PUR-HMI		$5,500.00		
Purchase-PLC Equipment	PUR-PLC		$27,000.00		
Purchase-VSD's & Softstarters	PUR-VSD		$4,000.00		
Subcontract M.C.C. & P.F.C.	SUB-MCC		$42,000.00		
Total Hardware Costs			**$86,500.00**		**25.33**
Electrical site expenses	SIT-EXP		$20,000.00		
Electrical site travel time	SIT-TRA	93.75	$7,500.00		
Subcontract Installation	SUB-INS		$126,000.00		
Total Instalation Costs		**93.75**	**$153,500.00**		**44.95**
Electrical Warranty Expenses	WAR-ENG		**$6,000.00**		**1.76**
Total Costs			**$341,500.00**		

Figure 68 - Project #2 Budget

Figure 68 shows the Budget allocated for second Plate Freezer project.

Project #3 - Refrigeration System					
Description	**Task Code**	**Budgeted Resource Hours**	**Budgeted Costs**	**Engineering Cost (%)**	**Total Costs (%)**
Administration-Budgeting & Mgt	ENG-ADM	141.45	$11,317.10	5.76	
Engineering & Drawing Time	ENG-DWG	318.77	$25,487.27	12.97	
Engineering Travel & Other Expenses	ENG-EXP		$16,040.00	8.16	
Factory Acceptance Testing	ENG-FAT	**62.29**	**$4,947.41**	2.52	0.91
Functional Descriptions	ENG-FD	**140.66**	**$11,253.59**	5.73	2.08
HMI Programming -SCADA & Touch	ENG-HMI	714.93	$57,165.14	29.08	
Electrical Manuals	ENG-MAN	112.11	$8,969.19	4.56	
PLC Programming	ENG-PLC	**492.38**	**$39,400.59**	20.04	7.28
Electrical Site Commissioning	SIT-COM	274.78	$21,982.38	11.18	
Total Engineering Costs		**2257.37**	**$196,562.67**		**36.30**
Purchase-Field devices or othr	PUR-FLD		$31,363.27		
Purchase-HMI Equip-includes PC	PUR-HMI		$10,695.57		
Purchase-PLC Equipment	PUR-PLC		$42,626.70		
Purchase-VSD's & Softstarters	PUR-VSD		$87,408.00		
Subcontract M.C.C. & P.F.C.	SUB-MCC		$17,695.08		
Total Hardware Costs			**$189,788.62**		**35.05**
Electrical site expenses	SIT-EXP		$45.00		
Electrical site travel time	SIT-TRA		$0.00		
Subcontract Installation	SUB-INS		$149,149.20		
Total Instalation Costs			**$149,194.20**		**27.55**
Electrical Warranty Expenses	WAR-ENG		**$5,934.07**		**1.10**
Total Costs			**$541,479.56**		

Figure 69 - Project #3 Budget

Figure 69 shows the Budget allocated for the first Refrigeration project.

Project #4 - Refrigeration					
Description	**Task Code**	**Budgeted Resource Hours**	**Budgeted Costs**	**Engineering Cost (%)**	**Total Costs (%)**
Administration-Budgeting & Mgt	ENG-ADM	133	$10,640.00	6.96	
Engineering & Drawing Time	ENG-DWG	216	$17,280.00	11.30	
Engineering Travel & Other Expenses	ENG-EXP		$42,273.12	27.65	
Factory Acceptance Testing	ENG-FAT	**62.5**	**$5,000.00**	3.27	0.93
Functional Descriptions	ENG-FD	**50**	**$4,000.00**	2.62	0.75
HMI Programming -SCADA & Touch	ENG-HMI	280	$22,400.00	14.65	
Electrical Manuals	ENG-MAN	40	$3,200.00	2.09	
PLC Programming	ENG-PLC	**280**	**$22,400.00**	14.65	4.19
Electrical Site Commissioning	SIT-COM	321.35	$25,708.40	16.81	
Total Engineering Costs		**1382.85**	**$152,901.52**		**28.59**
Purchase-Field devices or othr	PUR-FLD		$10,500.00		
Purchase-HMI Equip-includes PC	PUR-HMI		$3,000.00		
Purchase-PLC Equipment	PUR-PLC		$33,000.00		
Purchase-VSD's & Softstarters	PUR-VSD		$55,500.00		
Subcontract M.C.C. & P.F.C.	SUB-MCC		$125,000.00		
Total Hardware Costs			**$227,000.00**		**42.45**
Electrical site expenses	SIT-EXP		$6,714.00		
Electrical site travel time	SIT-TRA		$0.00		
Subcontract Installation	SUB-INS		$141,694.10		
Total Instalation Costs			**$148,408.10**		**27.75**
Electrical Warranty Expenses	WAR-ENG		**$6,454.81**		**1.21**
Total Costs			**$534,764.43**		

Figure 70 - Project #4 Budget

Figure 70 shows the Budget allocated for the second Refrigeration project.

Projects' Totals and Averages						
		Engineering Cost (%)				
Description	**Task Code**	**Project #1**	**Project #2**	**Project #3**	**Project #4**	**Average**
Administration-Budgeting & Mgt	ENG-ADM	7.42	9.42	5.76	6.96	**7.39**
Engineering & Drawing Time	ENG-DWG	12.03	12.57	12.97	11.30	**12.22**
Engineering Travel & Other Expenses	ENG-EXP	8.71	36.46	8.16	27.65	**20.24**
Factory Acceptance Testing	**ENG-FAT**	3.71	3.35	2.52	3.27	**3.21**
Functional Descriptions	**ENG-FD**	5.62	1.99	5.73	2.62	**3.99**
HMI Programming -SCADA & Touch	ENG-HMI	14.95	4.19	29.08	14.65	**15.72**
Electrical Manuals	ENG-MAN	3.37	2.20	4.56	2.09	**3.06**
PLC Programming	**ENG-PLC**	17.88	6.28	20.04	14.65	**14.71**
Electrical Site Commissioning	SIT-COM	26.31	23.54	11.18	16.81	**19.46**
				Engineering Costs		**100.00**
		% of the Total Costs				
Description	**Task Code**	**Project #1**	**Project #2**	**Project #3**	**Project #4**	**Average**
Factory Acceptance Testing	ENG-FAT	0.86	0.94	0.91	0.93	**0.91**
Functional Descriptions	ENG-FD	1.31	0.56	2.08	0.75	**1.17**
PLC Programming	ENG-PLC	4.16	1.76	7.28	4.19	**4.35**
				% of the Total Costs		**6.51**
		Total Costs (%)				
Description		**Project #1**	**Project #2**	**Project #3**	**Project #4**	**Average**
Engineering Costs		23.27	27.96	36.30	28.59	**29.03**
Hardware Costs		19.09	25.33	35.05	42.45	**30.48**
Instalation Costs		56.71	44.95	27.55	27.75	**39.24**
Warranty		0.93	1.76	1.10	1.21	**1.25**
				Total Costs		**100.00**

Figure 71 - Projects' Budget Average

Looking at the Figure 71, it can be seen that:

- Engineering costs accounts for 29.03 % of the Total Project Cost
- Hardware costs account for 30.48 % of the Total Project Cost
- Installation costs account for 39.24 % of the Total Project Cost
- Warranty accounts for 1.25 % of the Total Cost

In addition, it is important to note that:

- Functional Description accounts for 3.99 % of the Engineering cost and 1.17 % of the Total Project Cost
- PLC Programming accounts for 14.71 % of the Engineering costs and 4.35 % of the Total Project Cost

- Factory Acceptance Testing accounts for 3.21 % of the Engineering cost and 0.91 % of the Total Project Cost

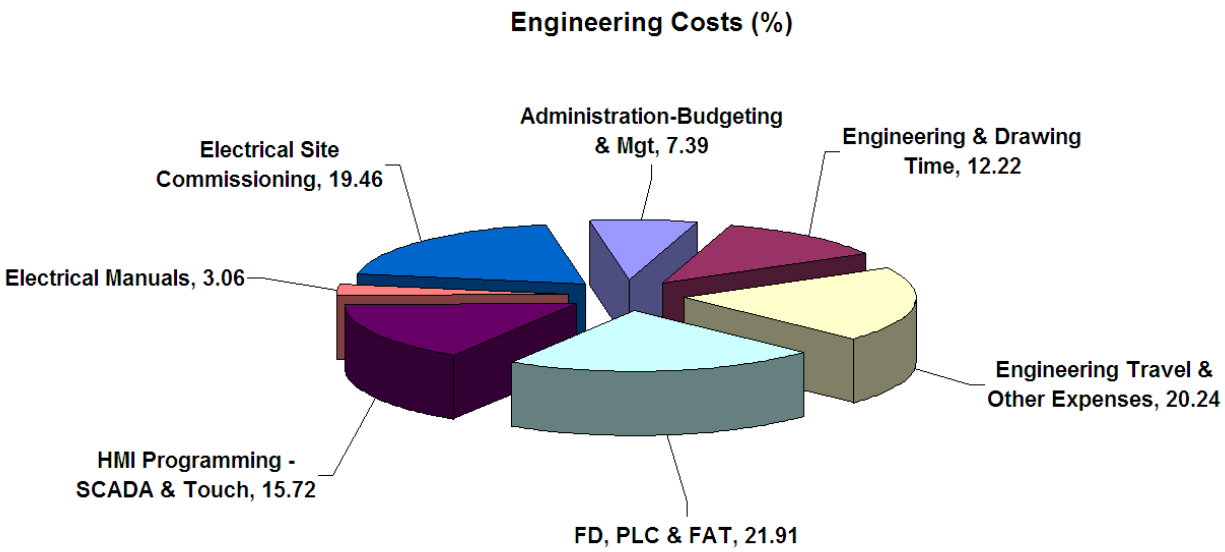


Figure 72 - Engineering Costs Graph (Average)

An area, which the APCG software can influence, amounts to 21.91% of the Engineering Cost, as shown in Figure 72, and 6.51% of the Total project, as shown in Figure 73.

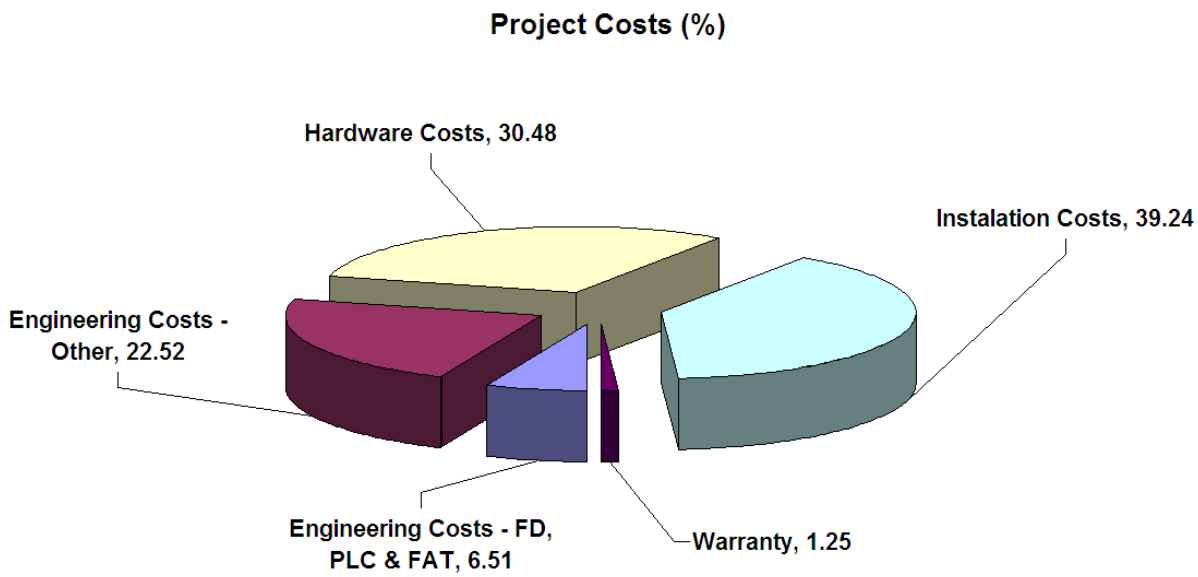


Figure 73 - Projects Total Costs (Average)

5.2 Real Life Project

The APCG software was used in one Plate Freezer project undertaken by Realcold Milmech Pty. Ltd. The Functional Description and PLC code was created by using the APCG software. In addition, the quality of the created PLC code was checked during Factory Acceptance Testing. Time has been recorded to measure the effectiveness of the APCG software.

Figure 74 and Figure 75 shows the results of using APCG Software. The results indicate the following:

1. By using APCG software, the time required to create Functional Description, PLC Code and to test created PLC code has improved. Budgeted time for FD was improved by 11 hours or 22%. PLC Time was improved by 88 hours or 55.35%. Time required to test the created PLC code, took 13 hours less than the budgeted time, or 39.4%.
2. All of the time improvements directly influence the cost of the project. Costs budgeted for creating Functional Description was improved by NZ$880.00. Costs required for PLC code was improved by NZ$7, 040.00, while testing of the PLC code was improved by NZ$1, 040.00.
3. The budgeted cost of the FD design, the PLC coding and the FAT testing, in total, was NZ$19, 360.00 or 27.21% or of the Total Engineering Costs. By using APCG software the cost was reduced by NZ$8, 960.00 or 12.59 %.
4. For the project, the cost was improved by 2.93 % overall.

Project #1 - Plate Freezers - Using APCG software

Description	Task Code	Budgeted Hours	Actual Hours	Difference (Hr)	Budgeted Costs	Budgeted Costs	Actual Costs	Difference ($)
Administration-Budgeting & Mgt	ENG-ADM	66			$5,280.00			
Engineering & Drawing Time	ENG-DWG	107			$8,560.00			
Engineering Travel & Other Expenses	ENG-EXP				$0.00			
Factory Acceptance Testing	ENG-FAT	33	20	13	$2,640.00	$2,640.00	$1,600.00	$1,040.00
Functional Descriptions	ENG-FD	50	39	11	$4,000.00	$4,000.00	$3,120.00	$880.00
HMI Programming -SCADA & Touch	ENG-HMI	133			$10,640.00			
Electrical Manuals	ENG-MAN	30			$2,400.00			
PLC Programming	ENG-PLC	159	71	88	$12,720.00	$12,720.00	$5,680.00	$7,040.00
Electrical Site Commissioning	SIT-COM	234			$18,720.00			
Calculations		**242**	**130**	**112**		**$19,360.00**	**$10,400.00**	**$8,960.00**
			Percentage(%)	**Percentage(%)**		**Percentage(%)**	**Percentage(%)**	**Percentage(%)**
Total Engineering Costs			**53.72**	**46.28**	**$71,158.24**	**27.21**	**14.62**	**12.59**
Total Hardware Costs					**$58,362.11**			
Total Instalation Costs					**$173,388.78**			
Electrical Warranty Expenses	WAR-ENG				**$2,843.37**			
Total Costs					**$305,752.50**	**6.33**	**3.40**	**2.93**

Figure 74 – APCG software used in Plate Freezer project

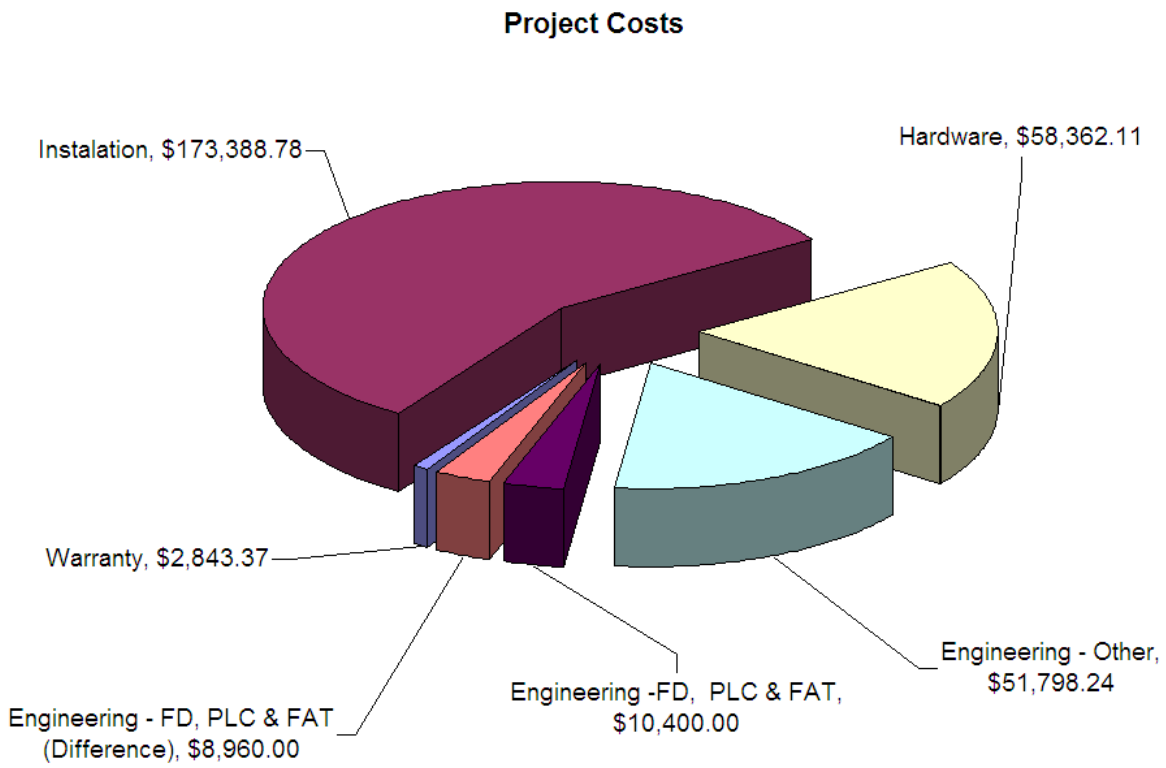


Figure 75 - APCG software used in Plate Freezer project (Graph)

5.3 Cost Implications

The APCG software was used in an ongoing project, undertaken by Realcold Milmech Pty. Ltd. The time required to design and test the PLC code was recorded. The records show that the APCG software improved the time required to design and test the PLC code, against budgeted time for the Test 2 Plate Freezer project.

Figure 76 shows that in the PLC programming area the time and cost was improved by more than 50 %. Time was also improved in designing a Functional Description, by using APCG software sequence tools.

The PLC code created with the use of APCG software is more reliable, and needs less testing time as it is shown in Figure 76.

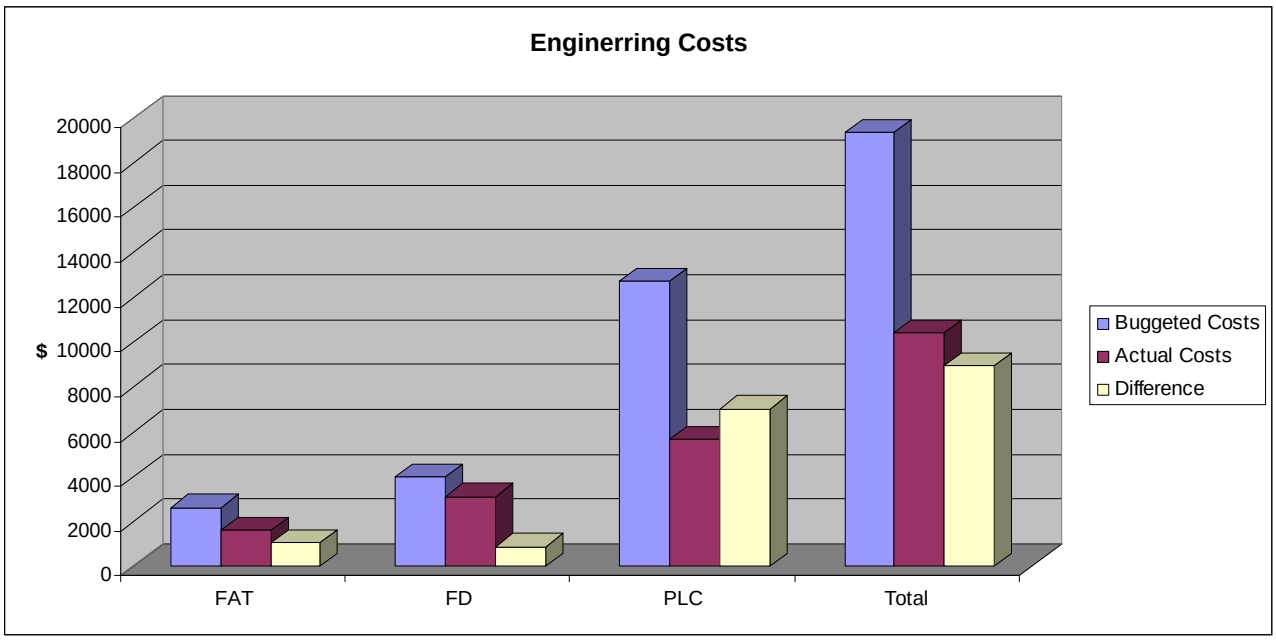


Figure 76 - Engineering Costs (Graph)

By using the results, shown in the Plate Freezer project, in the past projects it can be seen that there is a substantial savings in costs. Figure 77, Figure 78, and Figure 79 show the potential profit margins those projects would have if the APCG software was used for designing, programming and testing of the PLC code.

Project #2 - Plate Freezers			
	Budgeted Cost	Potential Savings	Savings %
Functional Descriptions	$1,900.00	$418.00	22.00
PLC Programming	$6,000.00	$3,321.00	55.35
Factory Acceptance Testing	$3,200.00	$1,260.80	39.40
Engineering Design	$95,500.00	$4,999.80	5.24
Total Project	$341,500.00	$4,999.80	1.46

Figure 77 - Project #2 Potential Savings

Project #3 - Refrigeration System			
	Budgeted Cost	Potential Savings	Savings %
Functional Descriptions	$11,253.59	$2,475.79	22.00
PLC Programming	$39,400.59	$21,808.23	55.35
Factory Acceptance Testing	$4,947.41	$1,949.28	39.40
Engineering Design	$196,562.67	$26,233.30	13.35
Total Project	$541,479.56	$26,233.30	4.84

Figure 78 - Project #3 Potential Savings

Project #4 - Refrigeration System			
	Budgeted Cost	Potential Savings	Savings %
Functional Descriptions	$4,000.00	$880.00	22.00
PLC Programming	$22,400.00	$12,398.40	55.35
Factory Acceptance Testing	$5,000.00	$1,970.00	39.40
Engineering Design	$152,901.52	$15,248.40	9.97
Total Project	$534,764.43	$15,248.40	2.85

Figure 79 - Project #4 Potential Savings

As shown above the profit margins differ from project to project. The profit margins can be anywhere from few thousands to tens of thousands.

6 Discussion

6.1 Research Contribution

The need to create an automatic PLC Code Generator for generating a fully functional PLC code has been recognized by identifying the shortcomings of the currently published literature and identifying a gap in the research. Automatic PLC code generation has been researched and partially accomplished according to the published literature [9]. In addition, research has been done in creating a blueprint on what an Automatic PLC Code Generator software should have and should be able to accomplish [1]. The majority of research has concentrated on automatically creating sequences [12], and the Body of the PLC code was not created.

A research "Automatic generation of PLC code beyond the nominal sequence" written by Guttel et al [1] indicates the structure of a PLC code. This research has been used as well as the PLC structure of the Realcold Milmech Pty. Ltd. in designing the APCG software in this thesis.

The objective of defining a PLC structure that the software mediator APCG will follow has been addressed. Under the guidance of the research, mentioned above, and the standard of the Realcold Milmech Ltd. Pty the APCG software was created. The APCG software has been designed to automatically create both parts of the PLC code, incorporating the sequence (the Brain) and the Body of the PLC code.

The Body of the PLC code incorporates structure for the following:

i. PLC Inputs and Outputs
ii. Alarms
iii. Motors
iv. Solenoids
v. HMI/SCADA monitoring
vi. Flags
    - Startup
    - Shutdown
    - Set up
    - Monitoring

The Brain of the PLC code incorporates structure for the following:

- Sequences in the Grafcet (step) format.

To accomplish the third objective, the MS Excel and VBA programs were used to create the APCG software. The MS Excel spreadsheets are used for user interface. The VBA (Visual Basic for Applications) is used to create a PLC code from the information entered on the MS Excel. The APCG software creates a PLC program, in an L5K file format. This file can be opened by a PLC program, the RSLogix5000 software.

The fourth objective was accomplished by using the APCG software in the Plate Freezer project, undertaken by Realcold Milmech Pty. Ltd. The Plate Freezer project had over 150 PLC Inputs and Outputs, with 90 sensors and 53 actuators.

The effectiveness and efficiency of the APCG software was recorded, as required in the definition of the fifth project objective. The time taken to design and test the PLC code for the Plate Freezer project was improved when using the APCG software in comparison to the budgeted time. The design of the Functional Description for the system was improved by 11 hours. To write a PLC code it took 88 hours less than budgeted, which is an improvement of more than 50 %. The PLC code created by APCG software was more reliable, which is proven by recorded time for the testing of the PLC code, which was 13 hours less than budgeted. Overall, Engineering costs, and thus, project costs were improved. The Engineering costs were improved by NZ$8, 960.00 or 12.59 %. The project costs were improved by 2.93 %.

For the last objective this thesis looked at past projects (section 5.1) and it can be concluded that the savings in cost would be NZ $4,999.80 for the second Plate Freezer project, NZ $26, 233.30 for the first Refrigeration project and NZ $15,248.40 for the second Refrigeration project.

The information above indicates that the research done has been successful, and all objectives were completed.

6.2 Limitations

There were a few factors, which restricted this research thesis.

- Number and variety of Real Life Projects - The projects done by Realcold Milmech are long in duration in relation to the time allocated for this thesis. One project can last up to 12 months. Consequently, it was difficult to get more than one Real Life project to test the efficiency of the APCG software.

- The Margin of Error for the cost improvements was unattainable because the APCG software was used in only one project.

- All of the AOI instructions were created from the authors experience while working for Realcold Milmech Pty. Ltd. As each company keeps their secrets to themselves, it was not possible to see how other companies have created their AOI instructions. Therefore, some of the information may not be agreeable for the setting of other companies.

6.3 Implications

This thesis has attempted to improve the efficiency of writing the PLC code. The need to be more competitive on the market is becoming greater[4]. Thus, any way of increasing the productivity has the potential to provide a company with a competitive edge. Based on the results of this project, the APCG software will be useful for programmers that tend to handle a variety of projects on a regular basis, where programming in a modular way is not appropriate.

Because more than a third of the budgeted time, for designing and testing a PLC code, is saved this has the potential to result in faster project turnover. This improved efficiency may lead to greater profit margin for the company as well as improved client satisfaction.

6.4 Future Research

This research can be considered as a second step in the creation of the APCG software. The first step was achieved by researches, such as Guttel et al [1], who have created a blueprint for this research.

The APCG software can be enhanced in number of ways:

- Selection of the Rockwell PLC Controllers – There are over 30 different controllers, which can be selected in RSLogix5000 software. At present, the APCG software works only with RSLogix Emulate 5000 Controller. The APCG software can be improved by adding more controllers to be available as a selection.
- Selection of the Rockwell PLC I/O cards – The APCG software does not connect the PLC code to the PLC I/O card. There is a wide selection of the PLC I/O cards for the Rockwell controllers. To enhance the APCG software further, a selection of PLC I/O cards can be added to it.
- Increasing the number of RSLogix5000 instructions – APCG software uses only the most common instructions used by the electrical engineers at Realcold Milmech Pty. Ltd. The number of PLC instructions is vast, and the additional AOI instructions make this number ever greater.
- Automatically generate HMI/SCADA database – Another opportunity to increase the effectiveness of the APCG software is to enable it to create HMI/SCADA database for various HMI/SCADA vendors.
- Upload the existing project into the Automatic PLC Code Generator software – To make some minor changes to the existing PLC code, as well as to keep the APCG software program up to date uploading existing projects into APCG software is an important part.
- Link two APCG software programs together, to create one PLC code – A large system could be split into sections that are more manageable for PLC programming. Each section could have its own APCG program. It would be beneficial if the APCG programs for each section can be linked to create the final PLC code for that system.
- Making the APCG software compatible, with other PLC types i.e. Siemens, Omron etc. – At present, the APCG software is only compatible with RSLogix5000.

7 Conclusion

The results of this project indicate that using APCG software is a more effective and efficient way of designing the PLC code than the traditional way. The APCG software was built according to the PLC program standard of Realcold Milmech Pty. Ltd. and the outline provided in the research paper by Guttel et al [1].

The APCG software was created by using MS Excel and VBA programs. MS Excel spreadsheets were used for the user interface, while VBA program was used to create the PLC code from the entered data.

Using, the APCG software, in a real life project it was apparent that it can be used to create a more reliable PLC program. In addition to this, the time taken to create the PLC program was greatly improved.

In conclusion, the APCG software can be used to create a fully functional PLC code, with minimised programming time and costs, thereby improving the profit margin for projects.

8 References

1. Knut Güttel1, P.W., Alexander Fay1, ***Automatic generation of PLC code beyond the nominal sequence***. Emerging Technologies and Factory Automation, 2008. ETFA 2008., 2008: p. 1277 - 1284

2. Waitakere Enterprise, *Get the Competitive Edge!*, W. Enterprise, Editor, Waitakere Enterprise.

3. http://en.wikipedia.org/wiki/Competition#Economics_and_business. *Competition*. Economics and business 2009.

4. ARC Advisory Group, *Programmable Logic Controller Worldwide Outlook.* 2009: p. 1.

5. Andy Dingley. *Programmable logic controller*. 2009 [cited 2009; History of the PLC controllers]. Available from: http://en.wikipedia.org/wiki/Programmable_logic_controller.

6. General Switchgear & Controls T&D Switches. *Control Metering & Relay Panels*. [cited 2009; Relay Based MCC]. Available from: http://www.gs.on.ca/control,metering_&_relay_panels.htm.

7. PAControl.com. *Programmable Logic Controller PLC*. [cited 2009; Basic Elements of a PLC controller]. Available from: http://www.pacontrol.com/PLC.html.

8. Zilio, D.C., S. Lightstone, and G.M. Lohman. *Trends in automating physical database design*. 2003: IEEE International Conference

9. Lee, H.-S.P.D.B.H.A.G.-B., ***Development for automatic control system***. Strategic Technologies, 2008. IFOST 2008., 2008: p. 421 - 424

10. Rockwell Automation Inc., *RSLogix5000 Controllers Add-On-Instruction*, Rockwell Automation Inc., Editor. 2008, Rockwell Automation Inc.,.

11. Leucht, K.W. and G.S. Semmel. *Automated Translation of Safety Critical Application Software Specifications into PLC Ladder Logic*. in *Aerospace Conference, 2008 IEEE*. 2008.

12. Thapa, D., et al., *Auto-Generation of IEC Standard PLC Code Using t-MPSG*. International Journal of Control Automation and Systems, 2009. **7**(2): p. 165-174.

13. KronoTech. *Automatic Code Generation*. 2009 [cited 2009 June 2009]; Automatic Code Generation - PLC Skeleton code]. Available from: http://www.kronotech.com/index.htm.

14. www.plccodegenerator.com. *PLC Code Generator*. 2009 [cited 2009 June 2009]; Available from: www.plccodegenerator.com.

15. Rockwell Automation Inc, *RSLogix Emulate 5000 Getting Results* Rockwell Automation Inc, Editor. 2004 Rockwell Automation Inc,.

16. Jack, H., ***Automating Manufacturing Systems*** *with PLCs*. 2007, 2008 Hugh Jack GNU Free Documentation License. p. 1-839.

17. Realcold Industrial, *Realcold - PLC Standards* in *Rev A* 2003, Realcold Industrial. p. 1-7.

18. Rockwell Automation Inc, *1769 SERIES Compact I/O Selection Guide*, Rockwell Automation Inc, Editor. 2007.

19. Rockwell Automation Inc., *Logix5000 Controllers General Instructions*, in *PN 957955-22*. 2005, Rockwell Automation Inc.,.

20. Rockwell Automation Inc, *Logix5000 Controllers Import/Export*. 2008.

9 Bibliography

Bradley, J.C., Millspaugh, A.C. 2001, *Advanced Programming Using Visual Basic 6*, Irwin McGraw Hill, New York

Bradley, J.C., Millspaugh, A.C. 2002, *Programming in Visual Basic 6*, Irwin McGraw Hill, New York

Frig, A. 2005, *Programming Excel/VBA*, viewed 21 November 2008, <http://www.staff.city.ac.uk/~fring/ExcelVBA/index.html>

Frig, A. 2005, *Programming Excel/VBA Part II*, viewed 26 July 2008, <http://www.staff.city.ac.uk/~fring/ExcelVBA/index.html>

Green, M., *How to use your Excel Add-In functions in VBA*, viewed 10 September 2009, <http://www.fontstuff.com/vba/vbatut08.htm>

Karabatsos, J 2004, *GUI Computing Coding Standards*, viewed 03 May 2008, <http://www.gui.com.au/resources/coding_standards_print.htm>

Kyd, C. 2006, *Read a Text File with VBA, and Write the Text to Excel*, viewed 12 February 2009, <http://www.exceluser.com/explore/questions/vba_textcols.htm>

Kyd, C. 2005, Corporate VBA standards for excel users who program, viewed 03 April 2009, <http://www.exceluser.com/explore/vbastds.htm>

Jelen, Bill 2005, *Special Edition Using Microsoft Office Excel 2007*, Que Publishing, Indianapolis

Jelen, B., Syrstad T. 2004, *VBA and Macros for Microsoft Excel*, Sams Publishing, Indianapolis

Microsoft, Visual Basic Developer Centre, Microsoft, viewed 2008-2009, <http://msdn.microsoft.com/en-au/vbasic/default.aspx>

Rockwell, 2007, *Logix5000 Controllers Import/Export Reference Manual, viewed 20 September 2008,* <http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm084_-en-p.pdf>

Rosa, V. 1999, *Writing Solid VBA Code*, viewed 17 June 2009, <http://www.geekgirls.com/vba_techniques.htm>

Schmidt, G. 2007, *Grafcet,* Festo Didactic GmbH & Co., viewed 20 March 2008, <http://www.festo-didactic.com/ov3/media/customers/1100/548679_grafcet_en_leseprobe.pdf>

10 Appendices

10.1 Appendix A: An Empty L5K file

```
(********************************************
 Import-Export
 Version   := RSLogix 5000 v17.00
 Owner     := NMastilovich, Realcold Industrial
 Exported  := Thu Jul 16 17:57:44 2009

 Note:  File encoded in UTF-8.  Only edit file in a program
       which supports UTF-8 (like Notepad, not Wordpad).
********************************************)
IE_VER := 2.8;

CONTROLLER Masters_Empty (ProcessorType := "Emulator",
                  Major := 17,
                  TimeSlice := 20,
                  ShareUnusedTimeSlice := 1,
                  RedundancyEnabled := 0,
                  KeepTestEditsOnSwitchOver := 0,
                  DataTablePadPercentage := 50,
                  SecurityCode := 0,
                  SFCExecutionControl := "CurrentActive",
                  SFCRestartPosition := "MostRecent",
                  SFCLastScan := "DontScan",
                  SerialNumber := 16#0000_0000,
                  MatchProjectToController := No,
                  InhibitAutomaticFirmwareUpdate := 0)
        MODULE Local (Parent := "Local",
                  ParentModPortId := 1,
                  CatalogNumber := "Emulator",
                  Vendor := 1,
                  ProductType := 14,
```

```
            ProductCode := 53,
            Major := 17,
            Minor := 2,
            PortLabel := "RxBACKPLANE",
            ChassisSize := 4,
            Slot := 1,
            Mode := 2#0000_0000_0000_0001,
            CompatibleModule := 0,
            KeyMask := 2#0000_0000_0001_1111)
    END_MODULE

    TAG
    END_TAG

    PROGRAM MainProgram (MAIN := "MainRoutine",
            MODE := 0,
            DisableFlag := 0)
              TAG
              END_TAG

              ROUTINE MainRoutine
              END_ROUTINE
    END_PROGRAM

    TASK MainTask (Type := CONTINUOUS,
            Rate := 10,
            Priority := 10,
            Watchdog := 500,
            DisableUpdateOutputs := No,
            InhibitTask := No)
                MainProgram;
    END_TASK

CONFIG ASCII(XONXOFFEnable := 0,
```

```
                DeleteMode := 0,
                EchoMode := 0,
                TerminationChars := 65293,
                AppendChars := 2573,
                BufferSize := 82) END_CONFIG

CONFIG ControllerDevice END_CONFIG

CONFIG CST(SystemTimeMasterID := 0) END_CONFIG

CONFIG DF1(DuplicateDetection := 1,
            ErrorDetection := BCC Error,
            EmbeddedResponseEnable := 0,
            DF1Mode := Pt to Pt,
            ACKTimeout := 50,
            NAKReceiveLimit := 3,
            ENQTransmitLimit := 3,
            TransmitRetries := 3,
            StationAddress := 0,
            ReplyMessageWait := 5,
            PollingMode := 1,
            MasterMessageTransmit := 0,
            NormalPollNodeFile := "<NA>",
            NormalPollGroupSize := 0,
            PriorityPollNodeFile := "<NA>",
            ActiveStationFile := "<NA>",
            SlavePollTimeout := 3000,
            EOTSuppression := 0,
            MaxStationAddress := 31,
            TokenHoldFactor := 1,
            EnableStoreFwd := 0,
            StoreFwdFile := "<NA>") END_CONFIG

CONFIG ExtendedDevice END_CONFIG
```

```
CONFIG FaultLog END_CONFIG

CONFIG FileManager END_CONFIG

CONFIG ICP END_CONFIG

CONFIG PCCC END_CONFIG

CONFIG Redundancy END_CONFIG

CONFIG SerialPort(BaudRate := 19200,
                  Parity := No Parity,
                  DataBits := 8 Bits of Data,
                  StopBits := 1 Stop Bit,
                  ComDriverId := DF1,
                  PendingComDriverId := DF1,
                  RTSOffDelay := 0,
                  RTSSendDelay := 0,
                  ControlLine := No Handshake,
                  PendingControlLine := No Handshake,
                  RemoteModeChangeFlag := 0,
                  PendingRemoteModeChangeFlag := 0,
                  ModeChangeAttentionChar := 27,
                  PendingModeChangeAttentionChar := 27,
                  SystemModeCharacter := 83,
                  PendingSystemModeCharacter := 83,
                  UserModeCharacter := 85,
                  PendingUserModeCharacter := 85,
                  DCDWaitDelay := 0) END_CONFIG

CONFIG UserMemory END_CONFIG

CONFIG WallClockTime(LocalTimeAdjustment := 0,
```

```
                    TimeZone := 0) END_CONFIG


END_CONTROLLER
```

10.2 Appendix B: L5K file instructions

Each rung consists of two parts, the condition and the output[19]. The conditions are on the left hand side of the rung and the outputs are on the right hand side (Figure 80).

The flow of the rung goes from left to right[19].

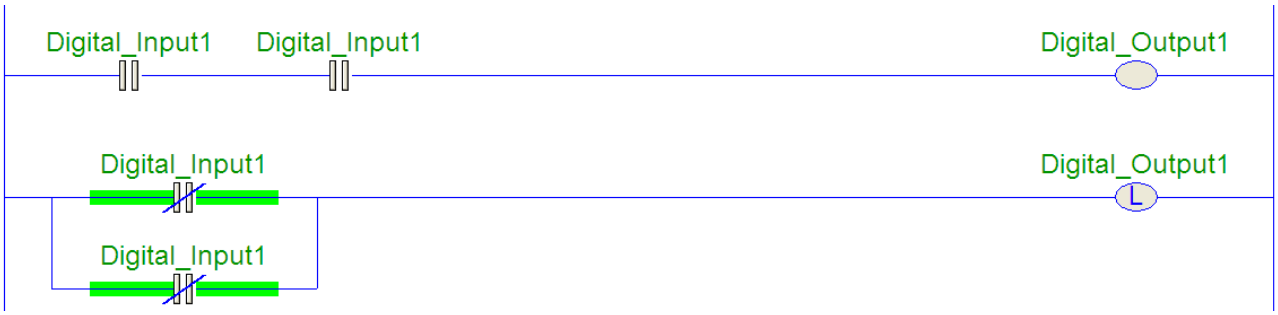


Figure 80 - Rung Flow

If the conditions are in series (top rung in Figure 80), that means that the condition 1 AND the condition 2 need to be true for the Output to turn ON.
If the conditions are parallel to one another (bottom rung in Figure 80) this means that condition 1 OR condition 2 needs to be true for the Output to turn ON.

XIC, XIO, ONS, OTE OTL and OUT Instructions

These instructions are bit instructions. These instructions are very common for Subroutine programming. All of the bit instructions are shown in Figure 81.

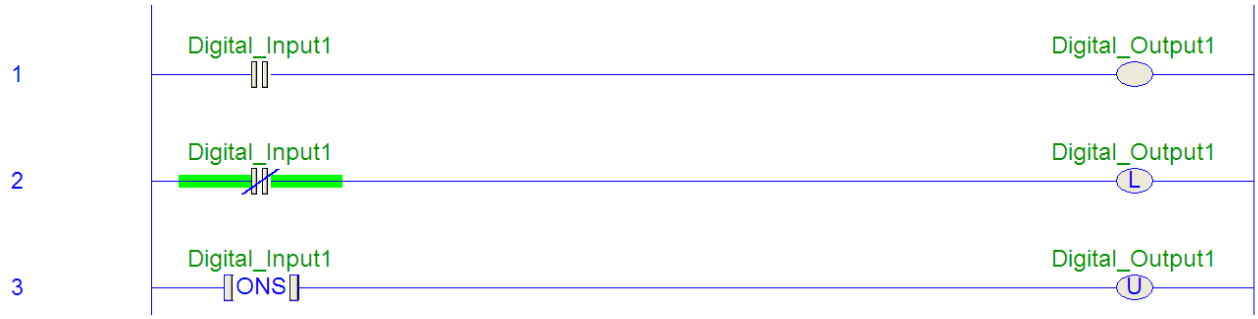


Figure 81 - Relay Subroutine

The L5K file representation of the instructions, shown in Figure 81 is as follows.

N: XIC(Digital_Input1)OTE(Digital_Output1);
N: XIO(Digital_Input1)OTL(Digital_Output1);
N: ONS(Digital_Input1)OTU(Digital_Output1);

XIC, XIO and ONS are condition instructions, while OTE, OTL and OTU are output instructions.
Each rung starts with the “N:” which indicates the start of the rung. The start of the rung is followed by the conditions than the output. Each rung is finished with the semicolon “;”.

Looking at Figure 80 and putting it into L5K file the code is represented as follow:

N: XIC(Digital_Input1)XIC(Digital_Input1)OTE(Digital_Output1);
N: [XIO(Digital_Input1) ,XIO(Digital_Input1)]OTL(Digital_Output1);

In Figure 80 first rung is simple one transition follows another. At the end of the rung, there is an output. The second rung is somewhat different, and has extra items associated with it. First thing that is different is the square bracket. The conditions, which are connected by an “OR” function, need to be put in the square brackets. In addition, it is important to notice that the “OR” function is represented by the comma.

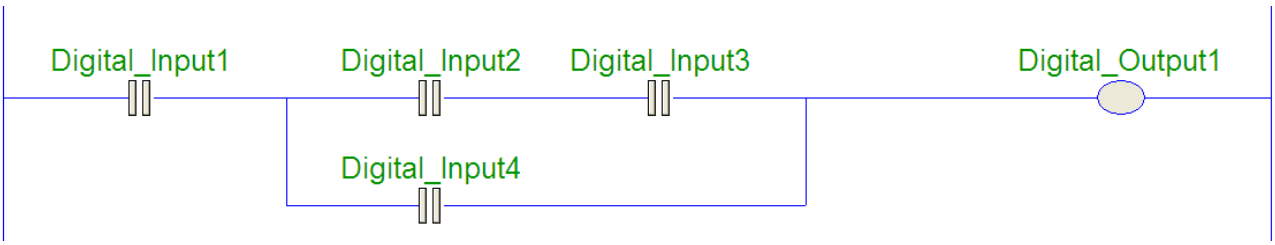


Figure 82 - Relay OR function

To explain the “OR” function a bit further, a more complex code is used, which is shown in Figure 82.

L5K file representation of the PLC code in Figure 82 is:

N:XIC(Digital_Input1)[XIC(Digital_Input1) XIC(Digital_Input1) ,XIC(Digital_Input1)]OTE(Digital_Output1);

As it was mentioned above, when conditions are linked with OR, a square bracket and comma need to be used.
When OR function is used the formula that needs to be used, for the Figure 82, is as fallows:

Condition1 AND [Condition2 OR condition3] = Output1

Condition 1 = Digital_Input1
Condition 2 = Digital_Input2 AND Digital_Input3
Condition 3 = Digital_Input4

So all the conditions that are part of the OR function need to be in the square brackets.

LES, LEQ, EQU, GEQ and GRT instructions

The instructions in Figure 83 are “Compare” instructions. For the compare functions, two integers (tags) are required, as one integer is compared against another.

LES – Less than instruction
LEQ – Less than or equal to instruction
EQU – Equal to instruction
GEQ – Greater than or equal to instruction
GRT – Greater than instruction

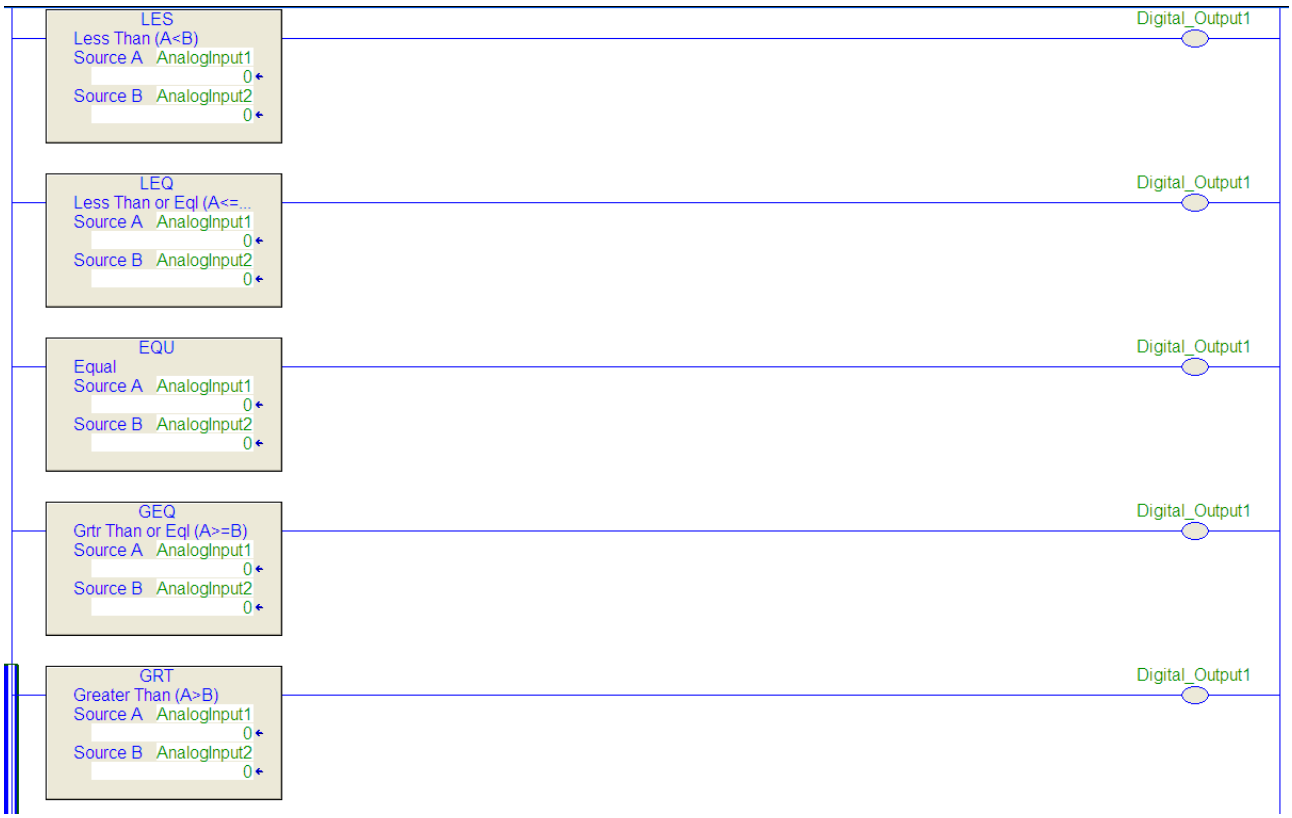


Figure 83 - Compare Instructions

L5K file representation of the PLC code in Figure 83 is:

N: LES(AnalogueInput1,AnalogueInput2)OTE(Digital_Output1);

N: LEQ(AnalogueInput1,AnalogueInput2)OTE(Digital_Output1);

N: EQU(AnalogueInput1,AnalogueInput2)OTE(Digital_Output1);

N: GEQ(AnalogueInput1,AnalogueInput2)OTE(Digital_Output1);

N: GRT(AnalogueInput1,AnalogueInput2)OTE(Digital_Output1);

MOV instruction

Move instruction (MOV) is another commonly used instruction, especially in Sequences. This instruction is used to move a value into an integer (tag).

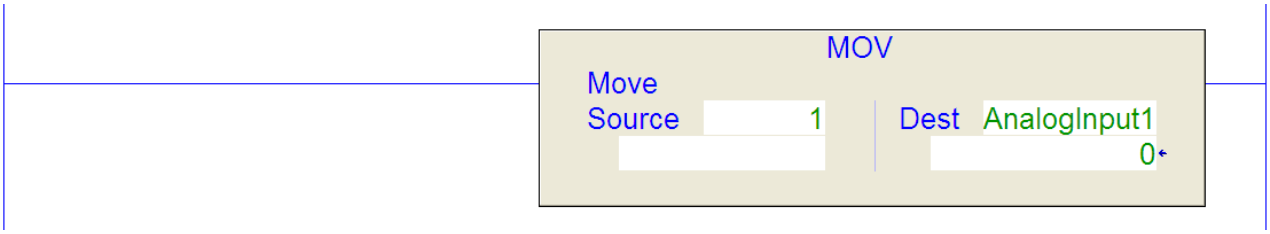


Figure 84 - Move instruction

This instruction also needs two tags, or one value and a tag to work. L5K file representation of the PLC code in Figure 84 is:

N: MOV(1,AnalogueInput1);

ADD, SUB, MUL and , DIV instructions

ADD – Add instruction
SUB – Subtract instruction
MUL – Multiply instruction
DIV – Divide instruction

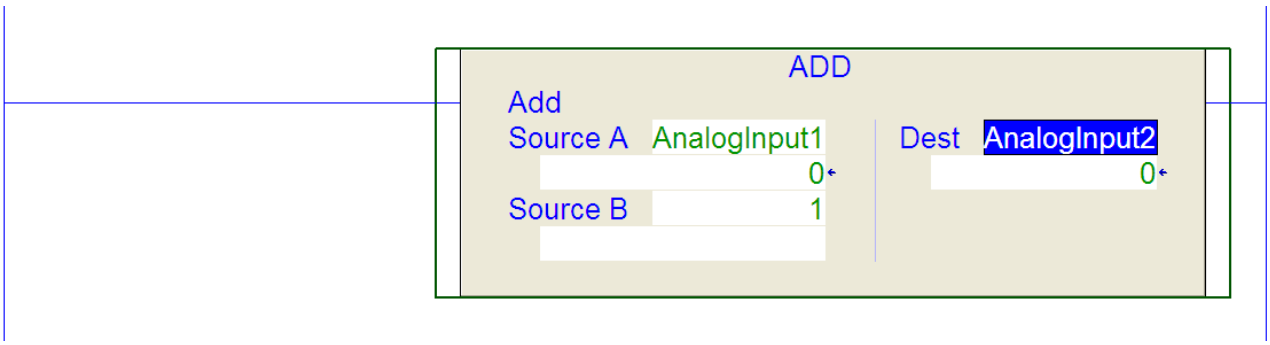


Figure 85 - Add instruction

L5K file representation of the PLC code in Figure 84 is:

N: ADD(AnalogueInput1,1,AnalogueInput2);

For these instructions, three tags or one value and two tags are required. Between each tag or value there is a comma.

TON, TOF, CTU and CTD instructions

Timers and counters are very important (and are always) part of the PLC code.

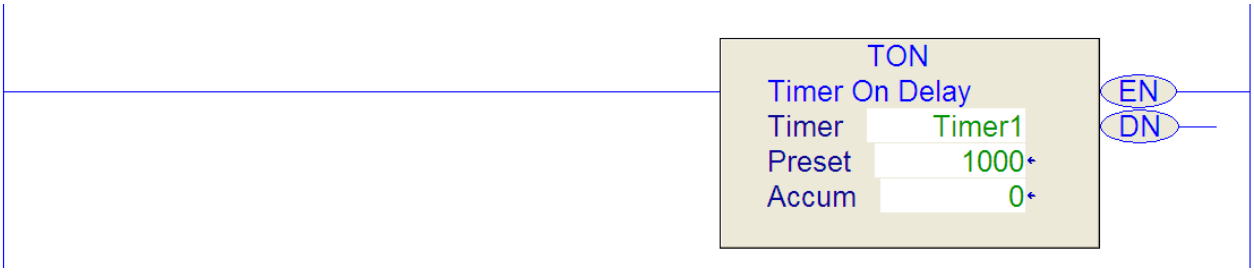


Figure 86 - Timer instruction

Each timer and counter has three fields that need to be filled, Timer tag, Preset value, and Accumulation value (Figure 86 and Figure 87).

Preset value is the set value, and timer counts to this value. Preset value of 1000 equates to 1 second. In the Figure 86, the timer is set to time up to one second. For the counters a reset instruction is used to reset the counter to zero.

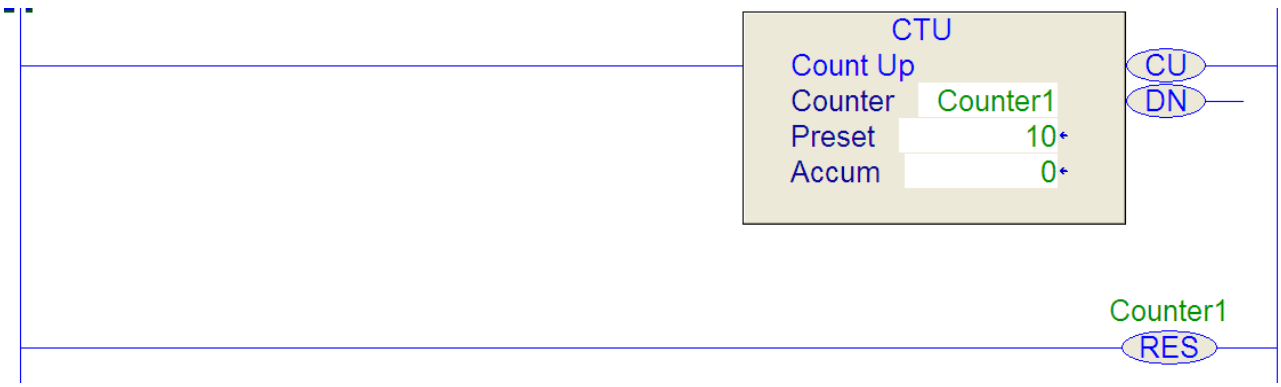


Figure 87 - Counter Instruction

The L5K representation of the code shown in Figure 86 and Figure 87 is as follows.

N: TON(Timer1,1000,0);

N: CTU(Counter1,10,0);

N: RES(Counter1);

Reset instructions can be used for timers as well.

10.3 Appendix C: Plate Freezer project PLC I/Os List

Plate Freezers' Common Sensors and Actuators	
Hydraulic Oil High Temp	Digital Sensor
Hydraulic Oil Low Level	Digital Sensor
Door 1 - safety switch	Digital Sensor
Door 2 - safety switch	Digital Sensor
Door 3 - safety switch	Digital Sensor
MCC E-Stop	Digital Sensor
Ceiling temperature sensor	Analogue Sensor
Wall temperature	Analogue Sensor
Solenoid Valve Pump Unloader	Solenoid
Scissor manifold circulation valve	Solenoid
Hydraulic power pack motor	Motor
Siren/Flashing Light	Actuator

Sensors and Actuators per Plate Freezer	
Note Items in Yellow are not in Plate Freezer #3	
Carton Stop 1 - up	Digital Sensor
Carton Stop 1 - down	Digital Sensor
Carton Stop 1 - blade down	Digital Sensor
Load conveyor 1 - extra carton	Digital Sensor
Load conveyor 1 - first carton	Digital Sensor
Load conveyor 1 - last carton	Digital Sensor
Load conveyor 1 - conveyor clear sender	Digital Sensor
Load conveyor 1 - conveyor clear receiver	Digital Sensor
Unload conveyor 1 - conveyor clear sender	Digital Sensor
Unload conveyor 1 - conveyor clear receiver	Digital Sensor
Stack 1 – in-feed clear sender	Digital Sensor
Stack 1 – in-feed clear receiver	Digital Sensor
Stack 1 – out-feed clear sender	Digital Sensor
Stack 1 – out-feed clear receiver	Digital Sensor

Stack 1 - plates level left	Digital Sensor
Stack 1 - plates level right	Digital Sensor
Stack 1 - Top stack fully open	Digital Sensor
Stack 1 - stack fully closed	Digital Sensor
Stack 1 - top left clamp cylinder extended	Digital Sensor
Stack 1 - top left clamp cylinder retracted	Digital Sensor
Stack 1 - bottom left clamp cylinder extended	Digital Sensor
Stack 1 - bottom left clamp cylinder retracted	Digital Sensor
Stack 1 - top right clamp cylinder extended	Digital Sensor
Stack 1 - top right clamp cylinder retracted	Digital Sensor
Stack 1 - bottom right clamp cylinder extended	Digital Sensor
Stack 1 - bottom right clamp cylinder retracted	Digital Sensor
Stack 1 - pecker cylinder left fully retracted	Digital Sensor
Stack 1 - pecker cylinder right fully retracted	Digital Sensor
Scissor 1 - home position	Digital Sensor
Stack 1 - height encoder	Analogue Sensor
Scissor 1 - encoder	Analogue Sensor
Plate freezer 1 - main lift raise	Solenoid
Plate freezer 1 - main lift lower	Solenoid
Plate freezer 1 - scissor extend	Solenoid
Plate freezer 1 - scissor retract	Solenoid
Plate freezer 1 - upper clamp extend	Solenoid
Plate freezer 1 - upper clamp retract	Solenoid
Plate freezer 1 - lower clamp extend	Solenoid
Plate freezer 1 - lower clamp retract	Solenoid
Plate freezer 1 - pecker extend	Solenoid
Plate freezer 1 - pecker retract	Solenoid
Plate 1 in-feed	Motor
Plate 1 out-feed	Motor
Carton Stop 1	Motor

In-feed Conveyors	
90 deg diverter - carton present	Digital Sensor

90 deg diverter - carton clear	Digital Sensor
90 deg diverter - motor home position	Digital Sensor
90 deg diverter - upstream accumulation full	Digital Sensor
Wall to diverter	Motor
Elliptical diverter	Motor

Out-feed Conveyors	
Out-feed Conveyor 1 - conveyor clear sender	Digital Sensor
Out-feed Conveyor 1 - conveyor clear receiver	Digital Sensor
Out-feed Conveyor 2 - conveyor clear sender	Digital Sensor
Out-feed Conveyor 2 - conveyor clear receiver	Digital Sensor
90deg transfer and decline	Motor
Out-feed to wall	Motor

LP Separator/Refrigeration	
LP separator Emergency Stop	Digital Sensor
LP separator - high level switch	Digital Sensor
LP separator - level differential pressure transducer	Analogue Sensor
LP separator - suction pressure transducer	Analogue Sensor
Pump 1 - discharge pressure transducer	Analogue Sensor
Pump 2 - discharge pressure transducer	Analogue Sensor
Pump - common pump suction transducer	Analogue Sensor
Plate freezer 1 - liquid solenoid valve	Solenoid
Plate freezer 2 - liquid solenoid valve	Solenoid
Plate freezer 3 - liquid solenoid valve	Solenoid
Liquid make up solenoid valve	Solenoid
Liquid make up modulating valve	Modulating Valve
Liquid Pump 1	Motor
Liquid Pump 2	Motor

10.4 Appendix D: CD Content

- Empty APCG program
- Empty RSLogix5000 l5k file
- Empty RSLogix5000 ACD file
- Real Life project - APCG program (only partial)
- Real Life project - RSLogix5000 l5k file (only partial)
- Real Life project - RSLogix5000 ACD file (only partial)