Conversion of SA30 PLC Program

Bachelor Thesis

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH School of Engineering at Campus Helsingborg
Industrial Electrical Engineering and Automation

Bachelor thesis:
Andreas Ahlqvist

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2013

Abstract

The objective of this thesis is to find the best possible solution to convert a program, Straw Applicator 30, from RSLogix 5000 to TwinCAT 3 with maintained functionality.
The main reason why Tetra Pak is interested in this problem is if the machine could be run with two different suppliers and what that requires. The conversion is briefly made rung by rung, except for where boxes appears that TwinCAT 3 got different in their library than RSLogix 5000 and those are converted with the requested functionality for the machine and not totally converted.

Keywords: RSLogix 5000, TwinCAT 3, Straw Applicator, Conversion

Sammanfattning

Målet med detta examensarbete är att hitta bästa möjliga lösning till att konvertera ett program, Sugrörsapplikator 30, så att samma funktionalitet som i RSLogix 5000 uppnås i TwinCAT 3.
Huvudanledningen till att Tetra Pak vill få detta examensarbete gjort är för att undersöka ifall det är möjligt att konvertera samt vad som krävs för att man ska kunna utrusta maskinen med utrustning från två systemleverantörer Beckhoff och Rockwell.
Kortfattat är programmet konverterat rad för rad förutom där olikheter mellan programmen uppstått, det har då konverterats utefter behov i programmet och därmed inte fått ett exakt identiskt beteende.

Nyckelord: RSLogix 5000, TwinCAT 3, Sugrörsapplikator, konvertering

Foreword

In this thesis method for converting a PLC program and each programming environment specific functions from RSLogix 5000 to TwinCAT 3 were studied.
This bachelor thesis project was carried out from January 2013 to June 2013 at the department for Industrial Electrical Engineering and Automation with Mats Lilja as academic supervisor and examiner.

List of contents

1 Introduction

1.1 Background

Development towards the standard IEC 61131 has been important for many PLC companies. Implementing the standard will make it easier for the costumers as the difference between different programs will be less. To convert from one supplier to another will be much easier when the standard is implemented.
The latest IEC 61131-3 edition (the third) were published in February 2013 with the language included

- Ladder diagram (LD)
- Function block diagram (FBD)
- Structured text (ST)
- Instruction list (IL)
- Sequential function chart (SFC)

The main motivation to use TwinCAT 3 is to decrease the execution time of the PLC program. TwinCAT 3 is time based and not continuously based program as RSLogix 5000.
It is cheaper to use a Beckhoff PLC than Rockwell/Allen-Bradleys, the price for CPU-capacity is lower using a Beckhoff PLC.

This bachelor thesis will investigate if it is possible to convert PLC code to TwinCAT 3 and how it should be done since RSLogix 5000 have some specific functions that TwinCAT 3 lacks, such as manipulating timers, sending messages and reading/setting system values.
To identify these kinds of differences and solving it will make a conversion of another machine easier in the future.

1.2 Purpose

The main purpose is to get a Straw Applicator 30 implements by a Beckhoff system with the same functionality as an Allen-Bradley/Rockwell implementation. This requires hardware changes such as servo-axis, communication handlers and the PLC. The main focus in this thesis will be to get the I/O and the communication with the HMI and frequency converters to have the same functionality as in RSLogix 5000.

1.3 Project delimitations

The focus is to get the I/O system working and be able to either compare to RSLogix to see if the behaviour is identical or simulate on a simulator or on a real machine. If there will be a simulation on a real machine it needs to replace the Allen-Bradley/Rockwell equipment such as the frequency converters, servo drives and PLC to Beckhoff hardware.

1.4 Tetra Pak

Tetra Pak began in the early 1950s as one of the first packaging companies for liquid milk. Since then Tetra Pak have been one of the world’s largest suppliers of packaging systems for milk, fruit juices and drinks, and many other products.
In 1991, Tetra Pak expanded into liquid food processing equipment, plant engineering, and cheese manufacturing equipment. Today, 2013, it is the only international company in the world able to provide integrated processing, packaging, and distribution line and plant solutions for food manufacturing. Tetra Pak provides processing solutions within the five food categories: dairy, cheese, ice cream, beverage and prepared food.

1.5 Technical Background

1.5.1 TwinCAT 3

TwinCAT 3 is based on soft PLC, this means that a computer sends I/O information by some kind of communication such as Ethernet. This makes it possible to test the code directly with the computer. The maximum it could be run on is 80% of the CPU power, the rest is needed for the operating system such as windows to be able to run. This system is based on cycle based scans, that means if a scan is complete, windows will be able to use the rest of the time while it waiting for the next scan to execute. The different languages it supports is those IEC 61131-3 includes, C/C++ and Matlab/Simulink. With the included Visual Studio Shell C/C++ is not able to run, you need a Visual Studio with more features for this. You can program object orientated with the languages included in the IEC 61131-3 standard. The base time could be from 50µs up to 1ms, during which inputs are read, the program is executed and the outputs are set.

1.5.2 RSLogix 5000

RSLogix 5000 follows the IEC 61131-3 standard. It provides ST, LD, FBD and SFC editors [2]. It is a continuously based program, this means it will start the next scan when the previous has finished, without any pause. To be able to test code that is written with RSLogix 5000 you need an RSLogix Emulate 5000 and RSLinx Classic both provided by Rockwell. The version used in this thesis is version 20.

1.5.3 Straw Applicator 30

The SA30 (Straw Applicator 30) is controlled by a LC (Line Controller) and the machine are placed after the filling machine in the production line. It is able to queue packages for the SA30 because of the distance between itself and the filling machine. To get the straws into the machine there is a box with straws that lies together in a line, waiting for one by one being drawn up and into the machine.

1.5.4 Package types

The SA30 that is considered in this thesis can handle Tetra Brik Aseptic (TBA), Tetra Brik (TB), and Tetra Prisma Aseptic (TPA). The packages can be seen in Figure 1-3.
The TBA and TPA are available in all package size and shapes from 80-500 ml to run on the SA30.

1.5.5 Straw Types

There are four straw types' available straight straws, U-straws, sensory straws and telescopic straws. The most common straw is the U-shaped. All the straws are shown in the Figure 1-1 and the sensory straw is shown in action in Figure 1-2.
The sensory straw is unique with it four holes that makes the liquid flow in four directions in the persons mouth that gives a different experience compared to the other straws.
Telescopic straws can provide a straw that is longer than the package.

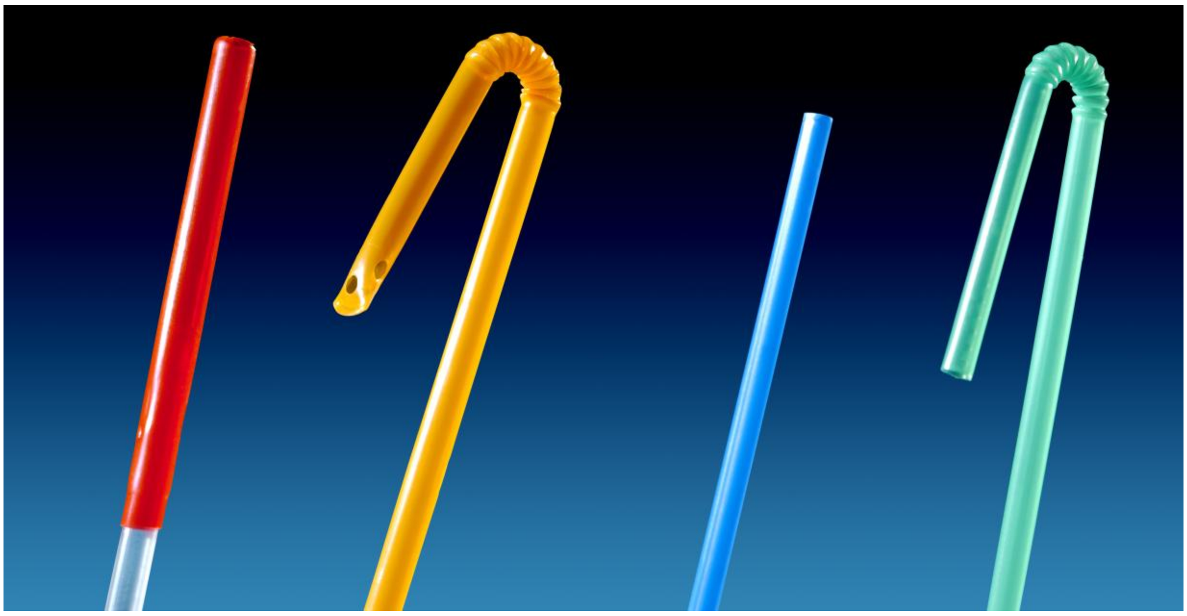


Figure 1-1

From left: Telescopic, Sensory, Straight & U-straw

(Tetra Pak, 2013)

Figure 1-2

Sensory straws in action

(Tetra Pak, 2013)

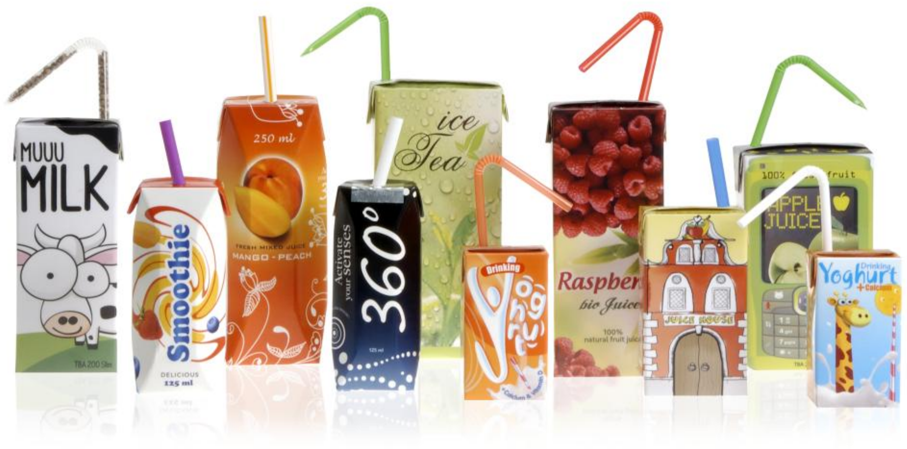


Figure 1-3
Tetra Brik Aseptic and Tetra Prisma Aseptic with straws
(Tetra Pak, 2013)

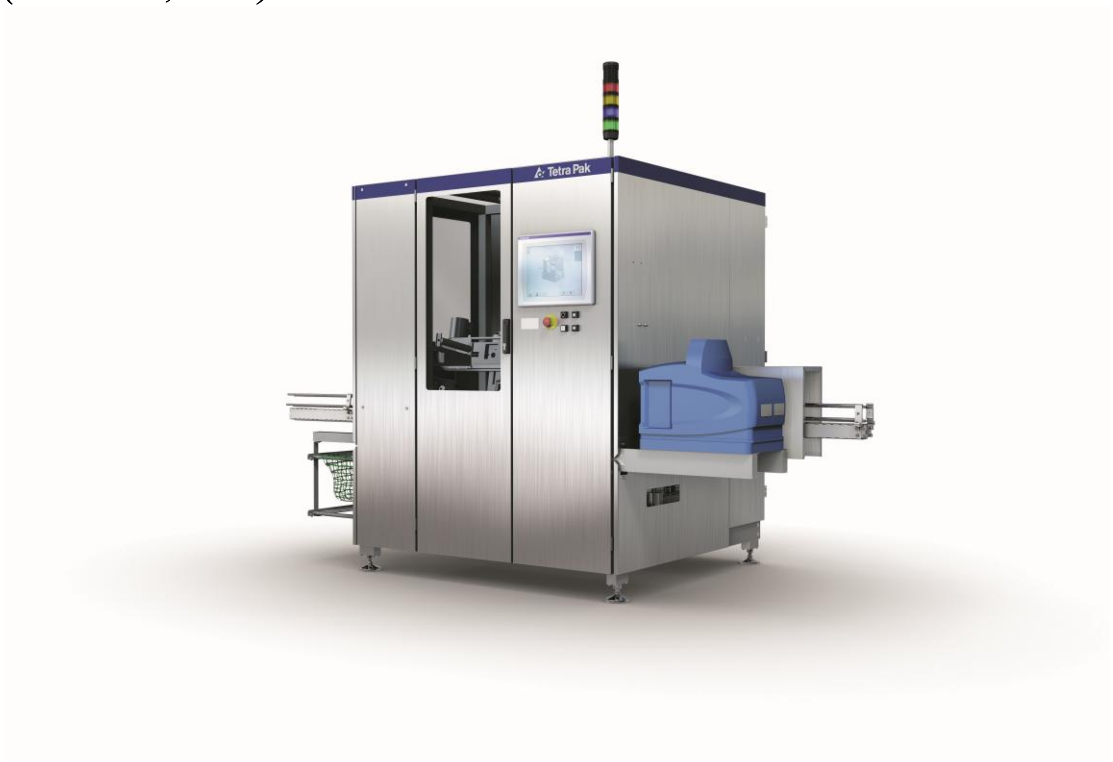


Figure 1-4
Straw Applicator 30
(Tetra Pak, 2013)

2 Methodology

2.1 Tetra Pak standard

Tetra Pak have its own standards to code a PLC program. The use of this Tetra Pak standard is intended to minimize the risk of non-human errors due to incorrect timing in the updating of the states.

2.2 IEC 61131-3

IEC 61131-3 is one of eight open international standards IEC 61131 for programmable logic controllers (PLC). The third edition was published in February 2013.

It consists of the following languages

- Ladder diagram (LD)
- Function block diagram (FBD)
- Structured text (ST)
- Instruction list (IL)
- Sequential function chart (SFC)

The main purpose of the standard is to easier get the same function of programs in different developing programs and also to easier test, debug and convert to other suppliers. The same function blocks, data types, functions, resources and programs will be used [1].

2.3 Structuring

Every program has its own routines in RSLogix 5000. The programs are very often modules that are sorted by what they are doing e.g. conveyor module, infeed module.

In TwinCAT 3, every POU (Program Organization Unit) has its own actions. Also every POU has its own states and transitions for SFC (Sequential function chart), that are visible in the ‘Solution Explorer’, where the actions also are visible.

3 Difference between RSLogix 5000 and TwinCAT 3

3.1 Timers

3.1.1 Structure

TwinCAT 3 timers have following structure

.IN	BOOL	Counts if it is true/false for a TON/TOF
.PT	TIME	How much it should count to
.Q	BOOL	If the timer is done
.ET	TIME	How much it have count to

IN and PT are inputs and ET and Q are outputs.
RSLogix 5000 timers have following structure

.EN	BOOL	Counts if it is true/false for a TON/TOF
.PRE	TIME	How much it should count to
.ACC	TIME	How much it have count to
.TT	BOOL	If its counting
.DN	BOOL	If the timer is done

EN and PRE are inputs, TT and DN are outputs and ACC is both in- and output.
Time in RSLogix 5000 is simply represented as a number expressed in milliseconds, to do it in TwinCAT 3 the time have to be specified with first T# that indicates that time (data type) is about to be used and then followed by numbers and then if its seconds or milliseconds, e.g. 10 seconds can be written as T#10s or T#10000ms.

3.1.2 Timer On Delay

The TON (Timer On Delay) instruction is a non-retentive timer that accumulates time when the instruction is enabled (rung-condition-in is true) (Rockwell, 2013).
A TON got a similar behaviour for TwinCAT 3 except for manipulating estimated time and reset that is used in this conversion.
To get a reset an explicit action needs to be implemented that sets the input false. The ET then returns to 0 and it will start counting from the beginning again when the input is true.

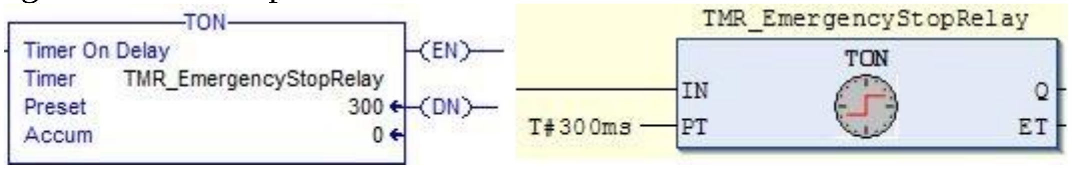


Figure 3-1

RSLogix 5000 TON (left) and TwinCAT 3 TON (right), both with the same functionality

3.1.3 Timer Off Delay

The TOF (Timer Off Delay) instruction is a non-retentive timer that accumulates time when the instruction is enabled (rung-condition-in is false) (Rockwell, 2013).
A reset command does not exist for TwinCAT 3, if a reset is required it needs specific user-created action that can be called which sets the input true.
Neither this nor the TON requires to be set to the rung in condition for counting. Next time the scan hits the timer it checks if it should be started and therefore the .IN does not need to be manipulated.

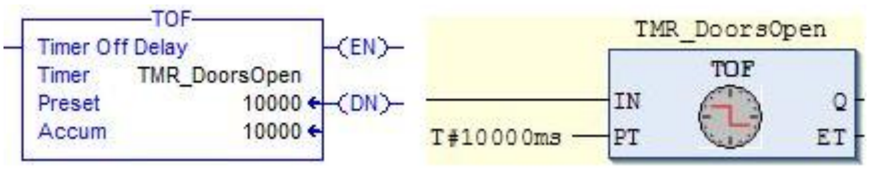


Figure 3-2
RSLogix 5000 TOF (left) and TwinCAT 3 TOF (right), both with the same functionality

3.1.4 Retentive Timer On

The RTO instruction is a retentive timer that accumulates time when the instruction is enabled. This simply means that the timer counts only when it is enabled and it remembers where it stopped counting and starts from there next time (Rockwell, 2013).
To reset the timer a reset instruction is needed. The output will remains true until the reset is executed.
This timer is only used once in this conversion.

To get a timer with the requested behaviour for TwinCAT 3 a TON timer is required with following code:

```
IF timer_Reset THEN
        timer_PT := presetTime;
END_IF

IF !rungConditionIn THEN
        timer_PT := timer_PT - timer_ET;
END_IF

timer(IN:=rungConditionIn);
```

3.2 Data Unit Type

The following DUT (Data Unit Type) are “dummies”, so that it should be able to compile without errors when such occur. Those are used for motion parts and that is not included in this thesis. These are DUT that are predefined in RSLogix which are used to control the system with. These will be explained and not used as it should, only used as “dummies”.

3.2.1 Message

MSG (Message) is mostly controlling the frequency converters which belong to the motion parts. A message is an instruction that asynchronous writes, or reads a block of data to another module on the network.
In this thesis this is not implemented in the code.

3.2.2 Copy File

The COP (Copy File) instruction copies the values in the source to the values in destination with the requested length without changing the source. The data types available are DINT, SINT, INT and REAL.
In this thesis this is not implemented in the code. This mostly appears in the motion parts.

3.2.3 Get/Set System Value

The GSV/SSV (Get System Value / Set System Value) instructions get and set controller system data that is stored in objects.
The data types that are available are INT, DINT, SINT and REAL for both GSV and SSV. To access a value a class name, instance name, attribute name and destination needs to be specified.

3.3 Coils

RSLogix 5000 has its own design possibilities where you could put a coil before a contact in the middle of a branch, an example is shown in Figure 3-3. In TwinCAT 3 this could be solved by splitting it up such as Figure 3-4 and Figure 3-5, so that the second branch starts with the output coil as contact and continue from there.

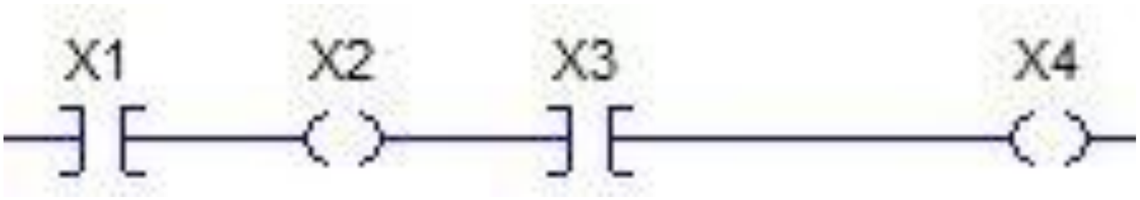


Figure 3-3
Coils after each other in RSLogix 5000

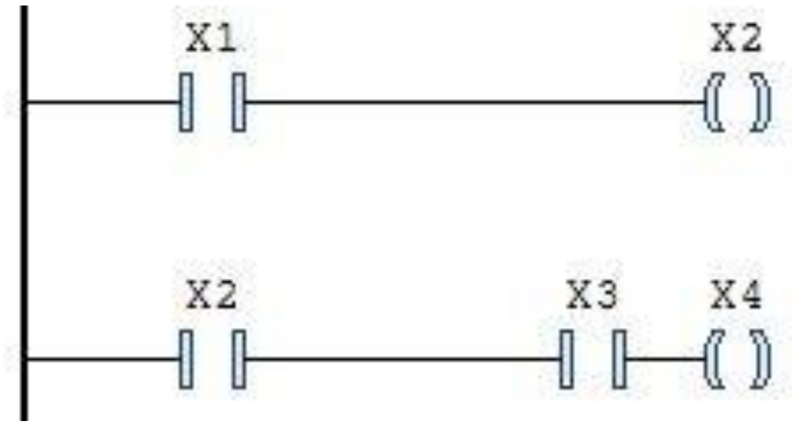


Figure 3-4

Same functionality as Figure 3-3 in TwinCAT 3

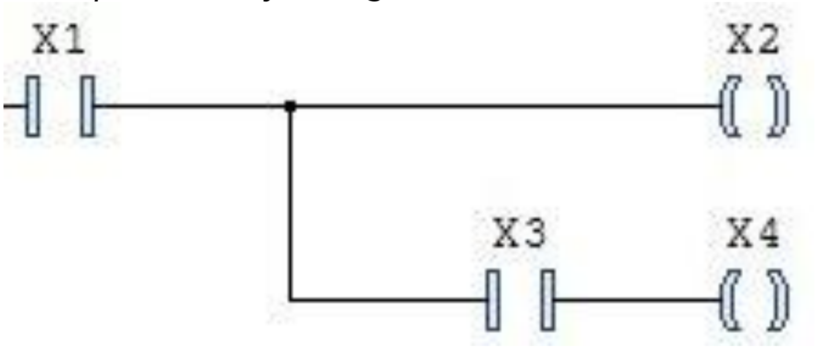


Figure 3-5

Same functionality as Figure 3-3 in TwinCAT 3

3.4 Declarations

A declaration of an array in RSLogix 5000 looks like ‘INT[100]’. This array of integers contains 100 elements. The same type of array is declared as ‘ARRAY OF INT[0..99]’ in TwinCAT 3. A declaration of a matrix has the same structure, but has the form [elements, elements] in RSLogix 5000 and [from..to, from..to] in TwinCAT 3.
A local variable in TwinCAT 3 is declared in the POU, which means that the variable is available for all the actions under the POU. The global variables are declared in GVL (Global Variable List).
The declaration of a local variable in RSLogix 5000 is done in Program Tags and is only local for the program and not the routines.

3.5 Rockwell specific functions

3.5.1 Bit access

By trying to access a bit by having the requested variable dot the requested bit generating “Bitaccess requires literal or symbolic integer constant” error, this works fine with RSLogix 5000, but not TwinCAT 3.
To access a specific bit 2^x is used where x is the requested bit.
e.g. (Decimal) 8 = (binary) 1000 = 2^3

3.5.2 Bit Field Distribute

The BTD (Bit Field Distribute) instruction copies the specified bits from the Source, shifts the bits to the appropriate position, and writes the bits into the Destination (Rockwell, 2013).

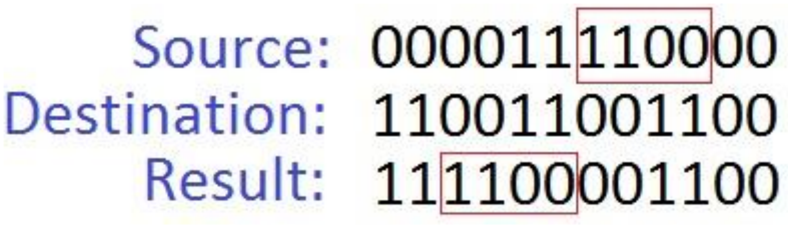


Figure 3-6
BTD instruction with following parameters, source bit 2, length 4, destination bit 6

```
FOR Counter := Source_Bit TO (Source_Bit + Length - 1) BY 1 DO
        DINTTemp := (Source AND
        LREAL_TO_DINT(EXPT(2,Counter))) OR DINTTemp;
END_FOR

DestTemp ≔ Dest;
Result := SHL(SHR(DestTemp, (Dest_Bit + Length + 1)), (Dest_Bit + Length
+ 1)) OR SHL(DINTTemp, Dest_Bit + 1) OR Dest AND
REAL_TO_DINT((EXPT(2,Dest_Bit + 1)-1));
```

DINTTemp stores the value that are about to be moved. TwinCAT 3 lacks any feature to accesses a specific bit. The method in 3.5.1 is used for bit access. To store the information without losses when accessing one bit at a time, DINTTemp is OR with the bit, so it stores in DINTTemp and only affecting one bit at a time.

To make sure that Dest have not been changed with the first use of it, it copies to DestTemp.
First SHR (Bitwise right-shift of an operand) the bits higher than those affected by DINTTemp to clear the bits that will be affected and those who are lower, then SHL (Bitwise left-shift of an operand) the same length to have cleared bits. By OR with the bits SHL that moves back the bits to right place and OR with the AND between only set bits with lower value than the moved bits and destination. As the code above shows.

3.5.3 Fill File

The FLL (Fill File) instruction fills elements of an array with the Source value. The Source remains unchanged (Rockwell, 2013).

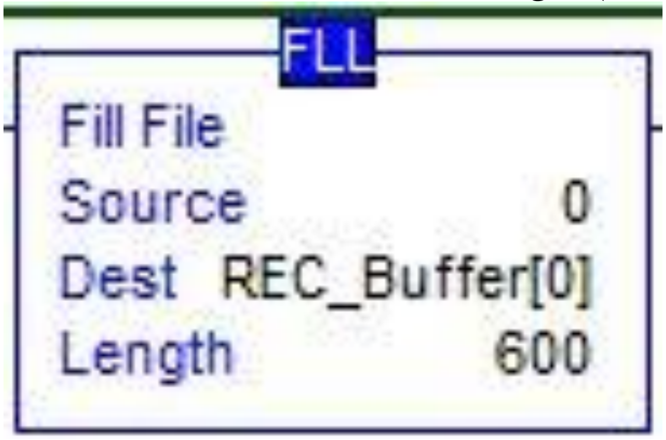


Figure 3-7
Fill File instruction in RSLogix 5000

To obtain the same behaviour in TwinCAT 3 as Figure 3-7 it could be done like:

```
FOR position TO 600-1 DO
        REC_Buffer[position] ≔ 0;
END_FOR
```

This requires a defined length to know how many elements that are about to be changed, therefore a unique function is needed every time with the array that are about to get information copied from the source.

3.5.4 Clear

The CLR (Clear) instruction clears all the bits of the Destination (Rockwell, 2013).
This can be made in TwinCAT 3 by using the 'MOVE' instruction to set 0 to the destination.
This could be executed in the beginning of a rung or an action to remove the present so no data is manipulated when beginning a new scan.
It could also be done in the end of a rung to clean up for the next action.

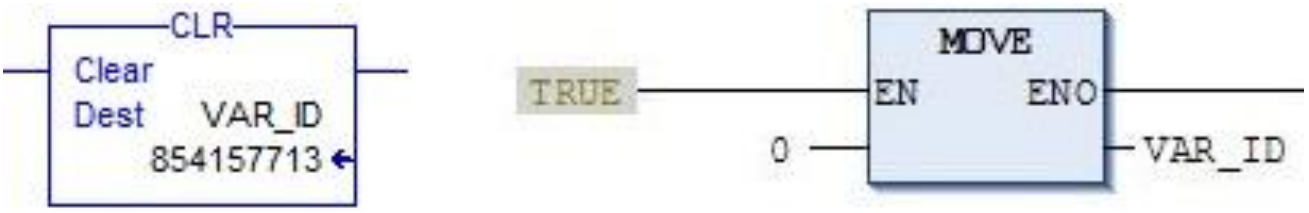


Figure 3-8
Clear instruction to the left in RSLogix 5000, the same logic to the right shown in TwinCAT 3

3.5.5 First scan

S:FS is a Bool in RSLogix 5000 that are only active during the first scan of the program. In TwinCAT 3 by declaring a FirstScan variable as Bool as true, this is done in every POU and at the end of the action where all other actions are called it sets to false and remains so until the program is restarted.
This function could activate units that need to be activated before the machine starts running.

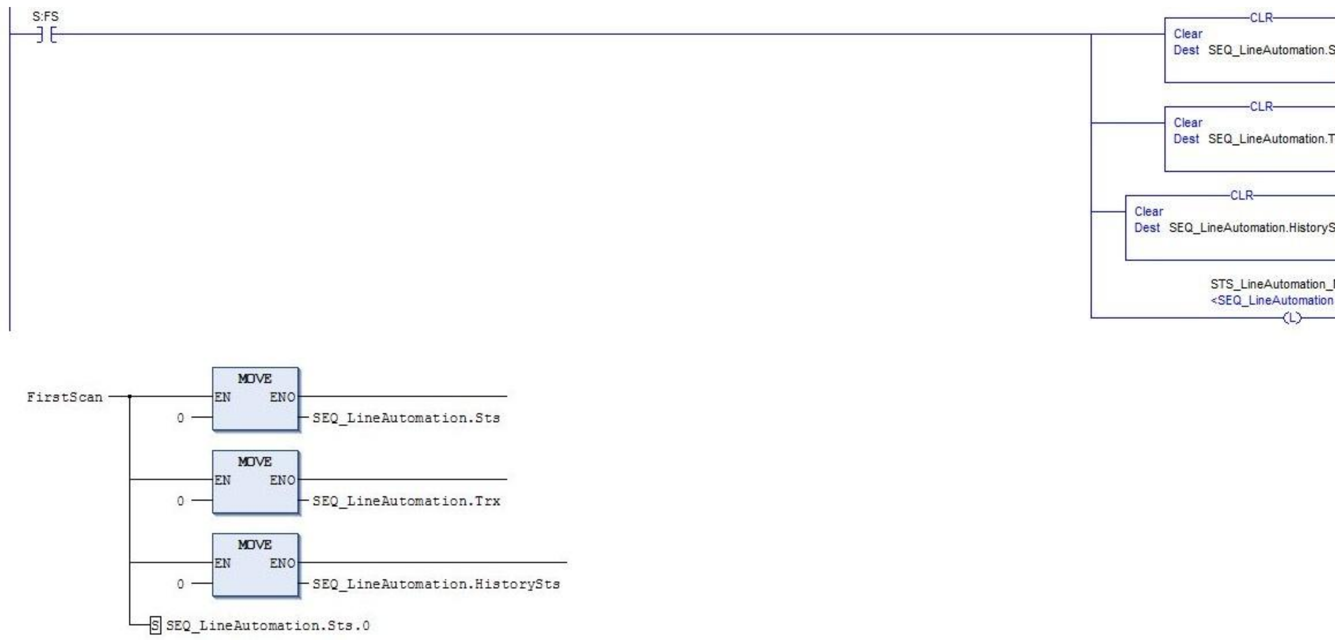
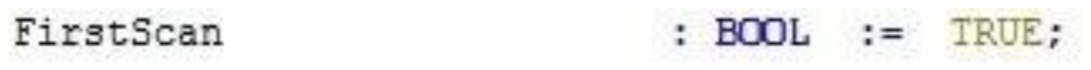


Figure 3-9
Typical use of FirstScan (S:FS) to clear transitions and states, the upper is RSLogix 5000 and the lower is TwinCAT 3



```
FirstScan                    : BOOL  :=  TRUE;
```

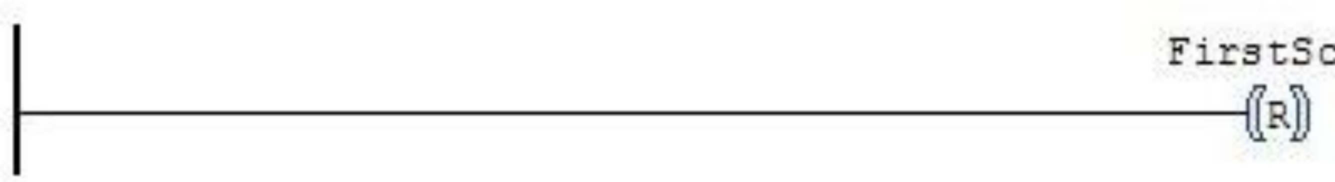


Figure 3-10
Declaration of FirstScan and Reset of it

3.5.6 Jump to Subroutine

The JSR (Jump to Subroutine) instruction jumps and executes the requested routine with input parameters if that’s required. To return from a SBR (Subroutine) the instruction RET (Return) is used.

To obtain this behaviour in TwinCAT 3 a box that refers to the SBR/action is added and to return a jump-action is used in the same way as in RSLogix 5000.

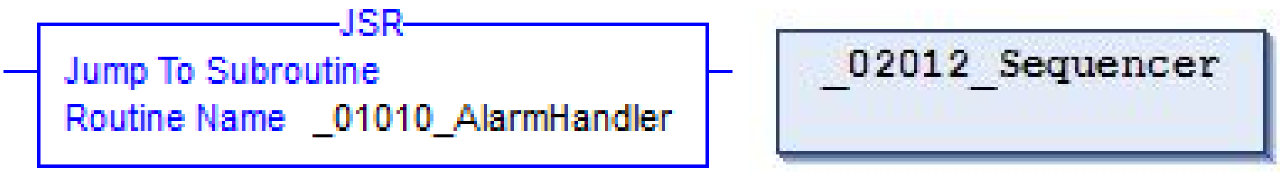


Figure 3-11
Jump to Subroutine instruction to the left in RSLogix 5000, the same logic to the right shown in TwinCAT 3

3.5.7 Assign task

Main routines in RSLogix 5000 is scanned every scan and from those it is possible to call the SBRs within the local programs.
To do the same in TwinCAT 3, every POU is assigned to “tasks”. By calling an action that is similar to the main routine in RSLogix 5000 the same behaviour can be emulated.

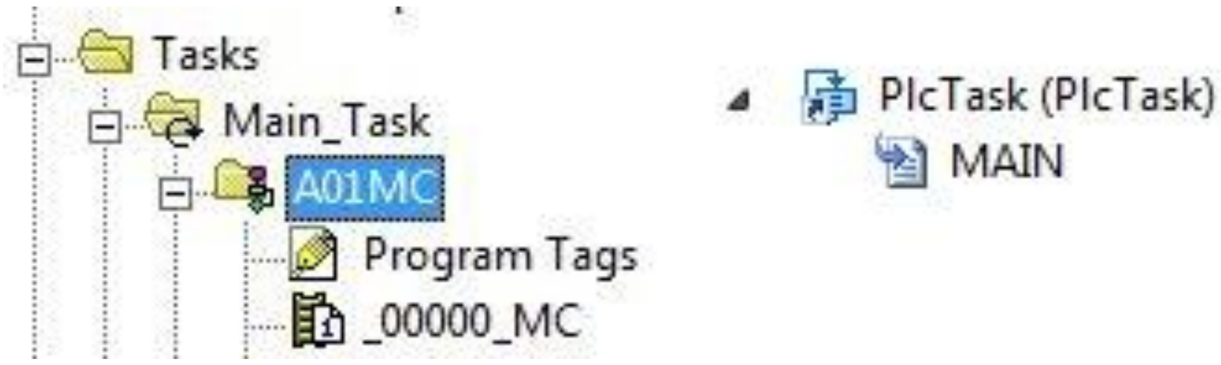


Figure 3-12
_00000_MC is an assigned task in RSLogix 5000 (right), MAIN is an assigned task in TwinCAT 3 (left)

3.5.8 One shot

When enabled and the storage bit is cleared, the ONS (One shot) instruction enables the remainder of the rung. When disabled or when the storage bit is set, the ONS instruction disables the remainder of the rung. (Rockwell, 2013)
This behaviour is able to get by using a rising edge detector in TwinCAT 3, where a falling edge detector also can be used to detect.

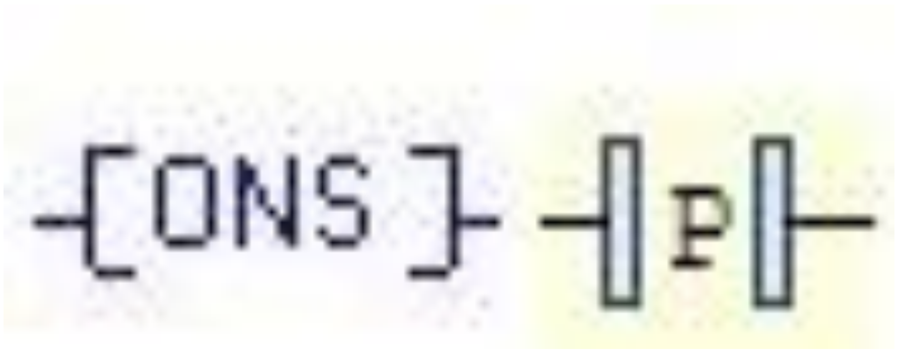


Figure 3-13
One Shot instruction in RSLogix 5000 to the left and TwinCAT 3 to the right

3.5.9 One shot rising

When enabled and the storage bit is cleared, the OSR (One shot rising) instruction sets the output bit. When enabled and the storage bit is set or when disabled, the OSR instruction clears the output bit (Rockwell, 2013).

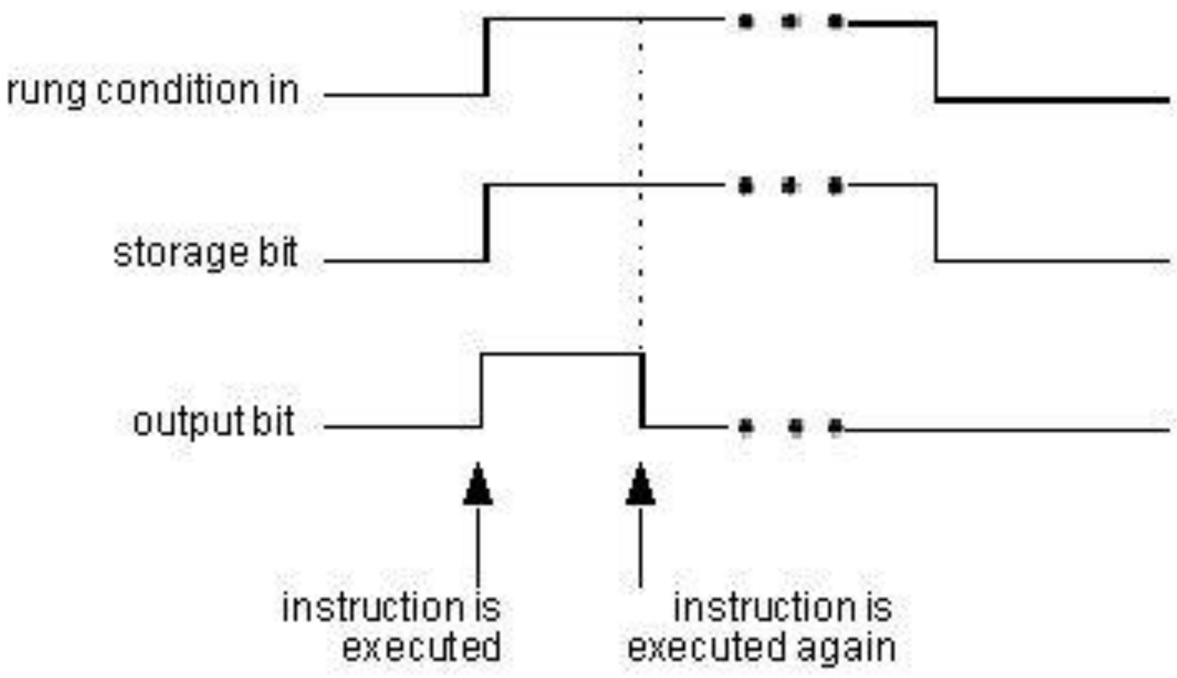


Figure 3-14
(Rockwell, 2013)
Explanation of the text above

To solve this for TwinCAT 3 one way is:

```
BOOLTEMP := FALSE;

IF inCondition=TRUE AND BOOLTEMP = FALSE THEN
        outCondition := FALSE;
        BOOLTEMP := TRUE;
```

```
END_IF

IF inCondition=TRUE AND BOOLTEMP = FALSE THEN
        outCondition := TRUE;
        BOOLTEMP := TRUE;
END_IF
```

3.6 Sequential function chart

In RSLogix 5000 it is possible to access a variable that tells if you are inside a step or if it is active and if it is the first time you scan it.
To solve this in TwinCAT 3, you could add entry and exit actions. By adding a variable on entry that sets true and gets false when leaving you can access if the current state is active. By adding another variable that sets true when entering and false at the end of the active state, you know if you're on the first scan.
When programming in RSLogix 5000 there are lines from every step to the possible steps to enter. In TwinCAT 3 there are jump-actions with where to go after those conditions that can go to other steps.

4 Discussion

TwinCAT 3 is a user friendly language, except for ladder logic that was full of graphical and logics errors. That made the programming more difficult then what it should have been.

4.1 TwinCAT 3

4.1.1 Ladder diagram

The Figure 4-1 shows the main reason why I used FBD instead of LD. It contains a graphical and logic error, where you are not able to have a parallel gate with a box at any time. By trusting the LD language, it could occur problems that are only visible when viewing the logic as FBD. By the time I realised this approximately one fourth of the program was converted and I had to check everything over again to find all the errors such as the Figure 4-1 shows. There was no problem only having parallel contacts, it was first when using boxes that the errors occurred.

At some times a “Codesys” error could occur, that Beckhoff GmbH also was unable to explain, see Figure 4-2 for an example. This only occurs in the beta and not the released version. Codesys is a development environment for PLC-programming according to PLCOpens IEC 61131-3 standard.

During this thesis I used TwinCAT 3.1.4002 to 3.1.4004 betas with build 4003 and when the conversion were finished 3.1.4006 was released.

Every error I explored was reported to Beckhoff GmbH.

Even though I used a beta and reported the error with parallel boxes in Figure 4-1 it still remains in the official version 3.1.4006.

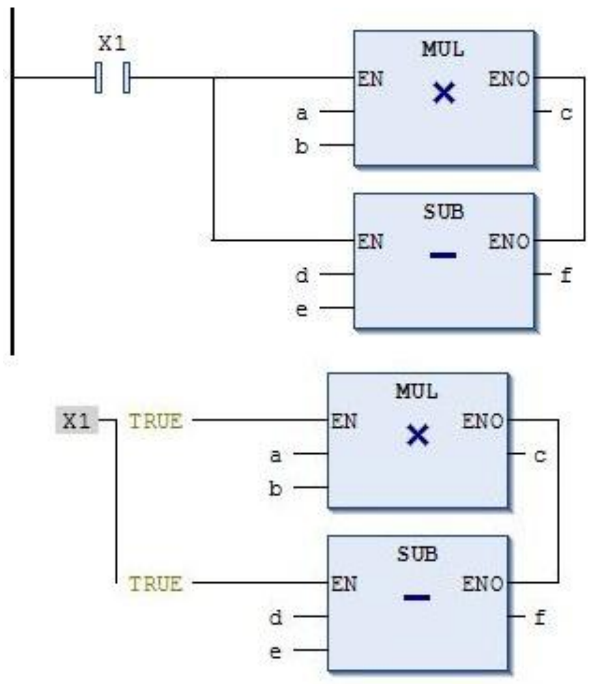
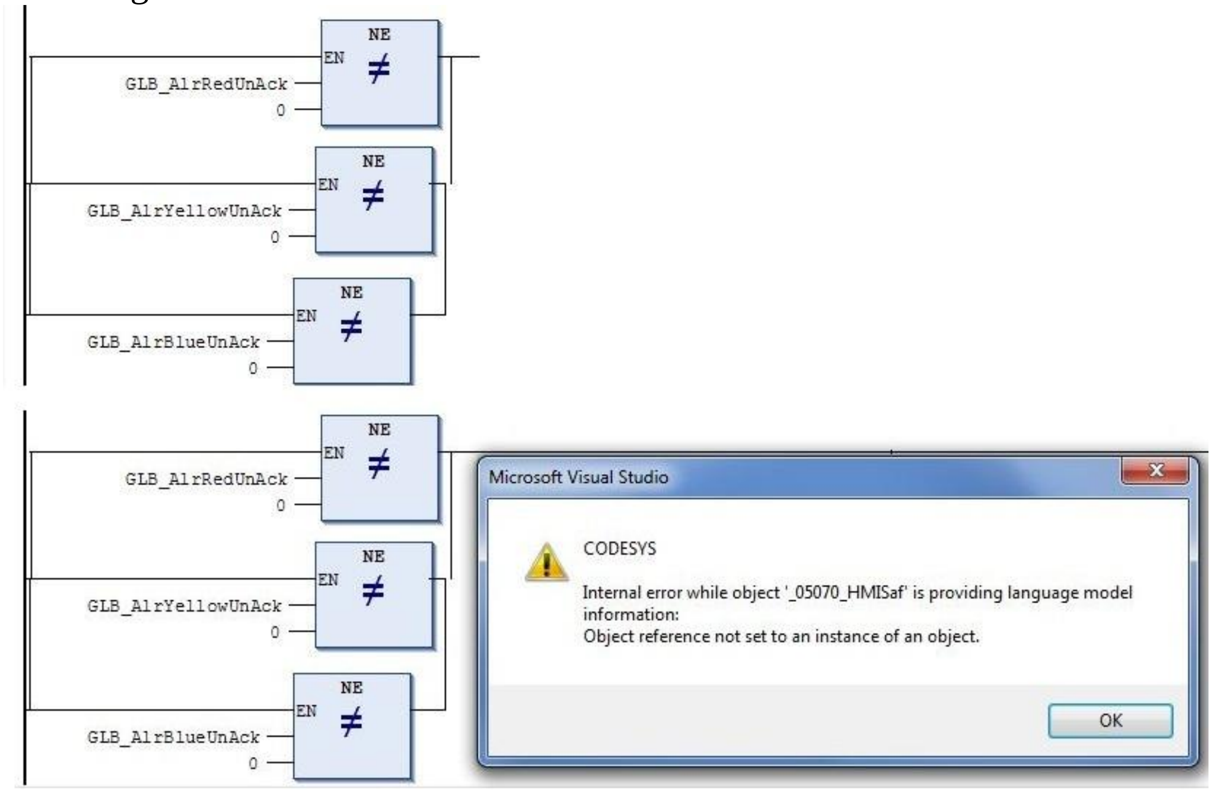


Figure 4-1
Parallel boxes in LD (top) and the same logic in FBD (bottom)

The graphical in Figure 4-1 LD view shows what to be a connected rung, when switching to FBD view there are two separate rungs, one with only input and no output, the other one that will always be executed because of the inputs will always be true. By trusting LD graphical problems could appears when switching to FBD view that are not visible in LD view.



CODESYS

Internal error while object '_05070_HMISaf' is providing language model information:
Object reference not set to an instance of an object.

Figure 4-2
Parallel boxes in the beginning of a rung that generates CODESYS

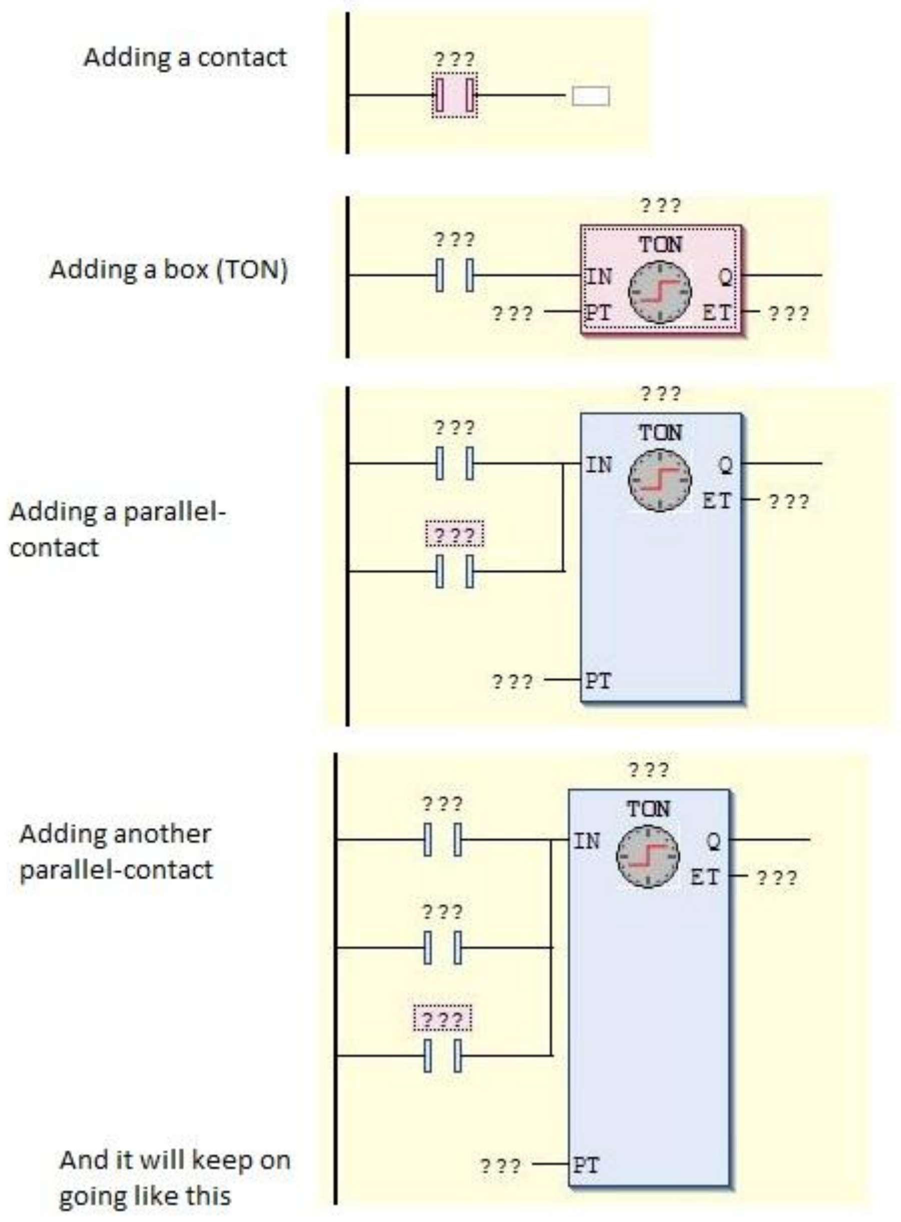


Figure 4-3
A box that gets bigger the more contacts it is in front of it.

The Figure 4-3 shows a graphical view with the standard preferences, where the boxes becomes larger and larger with the contacts added just before. This is able to solve in the preferences by widening the gap between the logical operands.

4.1.2 Sequential function chart

In the beginning this was annoying to use because of all the errors that occurred. Many errors were not possible to be explained either by me or Beckhoff GmbH. However, when redoing the action for 3 times it finally worked, without any good explanation. This language became easier with the time as experience was gained.

4.1.3 Structured text

Beckhoff is promoting ST (Structured text) having spent a lot of time using it. It was easy to use as it should be and not full of errors.
Accessing functions is done by first using the functions name and then what to do with the values, e.g. a timer called timer.
timer(time:=T#3ms)
This manipulates the time to count to 3ms. The ‘T’ means time, the # is to separate T from the value. 3 is the value in the unit after, in this case it is milliseconds. With s second is the unit.

4.1.4 Function block diagram

I found this very pleasant to work with after a while. It was a different way of thinking than what I was used to. But I found it very stimulating way of thinking when structuring a program. It started with the output and ended with the inputs when programming it.
The main reason why I started using FBD is that LD was not working as it should and therefore it was not possible to see if it was correctly programmed. See Figure 4-1for an example.

4.1.5 Assigning task

Assigning tasks was not working as it should have done, I was able to assign one task, not more than that, and from that task I called other actions that called other actions. This is an error not yet solved in the official version 3.1.4006 of TwinCAT 3. It only appeared one assigned task and when a new task was assigned it did not appear in the assigned task list.
If it would have worked as it should it would be possible to assign the POUs that calls the action within the POU.

4.2 Verification of SA 30

The conversion were about converting and not verifying, therefore it was not confirmed if the code worked on a physical machine. The way it was confirmed was by executing the program within TwinCAT 3.

The conversion was about converting branch by branch to obtain the same behaviour and not converting whole of actions or POUs for obtaining it.
By testing smaller parts of the code in both environments to obtain the same behaviour was made when TwinCAT 3 lacked functions that RSLogix 5000 contained.
The conversion does not guaranty that the motion parts will be able to fit with it, there will be a different time for executing the actions which needs a different cycle time, since RSLogix 5000 is cycle based and TwinCAT 3 are periodic based, the time will be calculated when the conversion of the machine is completed. The motors could be different designed with different input parameters that needs a unique configuration and also contain unique functions in RSLogix 5000 that TwinCAT 3 lacks, that the converted parts are adapted for and therefore needs to be either removed or changed.

4.3 Source criticism

The sources that are used are Rockwells internal help in RSLogix 5000, that is provided by Rockwell. This source is worth trusting since the provider of it is the provider of RSLogix 5000.
PLCopen is used because it supplies the IEC 61131 standards.

5 Conclusion

I did not find any research regarding conversion from RSLogix 5000 to TwinCAT 3. I was able to convert all of the parts of the program that I were about to convert and find solutions for all the differences that needed to be solved except for one. The one that I was not able to solve were about moving directly into a state in SFC from LD. Beckhoff GmbH was not able help me solving it either.
Since no fully functional SA 30 was available at the time, it was not feasible to assess the possibility to convert another machine, especially not with respect to the motion parts. If the machine works as it should when the motion parts are converted I think there is a big potential to have two suppliers for the machine and to be able to move freely between them.

By having two suppliers Tetra Pak will become more vendor independence. The conversion of the motion parts are not finished yet, but if they will be working as requested it will be possible to switch between the systems. This does not considering further updates to the machine, if they fit or not.

5.1 Further development

The conversion was very time consuming for a human to do. Since this was a rung by rung conversion there could be a program that does it all by itself. By exporting the program from RSLogix 5000 to a converter that converts it rung by rung into TwinCAT 3 would be very timesaving. There is no research made regarding a programmed converter.
By having this program fully functional the development could be made in one of the environments and be directly converted without doing the same thing again in the other environment.

References

1. http://www.plcopen.org/pages/tc1_standards/iec_61131_3/ 2013-06-09
2. http://www.rockwellautomation.com/rockwellsoftware/design/rslogix5000/overview.page 2013-06-10

Appendix I - List of abbreviations

SA30	-	Straw Applicator 30
POU	-	Program Organization Unit
DUT	-	Data unit type
PLC	-	Programmable logic controller
SFC	-	Sequential function chart
LD	-	Ladder diagram
FBD	-	Function block diagram
ST	-	Structured text
IL	-	Instruction list
TBA	-	Tetra Brik Aseptic
TPA	-	Tetra Prisma Aseptic
TB	-	Tetra Brik
TCA	-	Tetra Classic Aseptic
TWA	-	Tetra Wedge Aseptic
TT	-	Tetra Top
COP	-	Copy file
SSV	-	Set system value
GSV	-	Get system value
BTD	-	Bit field distribute
TON	-	Timer on delay
TOF	-	Timer off delay
RTO	-	Retentive timer on
MSG	-	Message
FLL	-	Fill file
OSR	-	One shot rising
CLR	-	Clear
INT	-	Integer
SINT	-	Short integer
DINT	-	Double integer
BOOL	-	Boolean
SHL	-	Bitwise left-shift of an operand
SHR	-	Bitwise right-shift of an operand
JSR	-	Jump to Subroutine
SBR	-	Subroutine
RET	-	Return
ONS	-	One shot
OSR	-	One shot rising
GVL	-	Global Variable List

Appendix II – Word List

Non-retentive – The time does not reset until a reset instruction is called.

Continuously based – Scans the program again directly after finishing the previous scan

Cycle/Time based – Scans the program on a time interval, will have a pause before next scan.