Ninth LACCEI Latin American and Caribbean Conference (LACCEI'2011), Engineering for a Smart Planet, Innovation, Information Technology and Computational Tools for Sustainable Development, August 3-5, 2011, Medellín, Colombia.

A PLC Automated Security Checkpoint

Marvin Blackman
Mechatronics Engineering, Vaughn College of Aeronautics and Technology, Flushing, NY, USA,
Email: marvin.blackman@gmail.com

Shahidul Islam
Mechatronics Engineering, Vaughn College of Aeronautics and Technology, Flushing, NY, USA,
Email: shahidul1118@yahoo.com

Joseph Kamel
Mechatronics Engineering, Vaughn College of Aeronautics and Technology, Flushing, NY, USA,
Email: joek2316@gmail.com

Advisor: Hossein Rahemi, Ph.D.
Professor and Chairman, Department of Engineering and Technology, Vaughn College of Aeronautics and Technology, Flushing, NY, USA, Email: hossein.rahemi@vaughn.edu

ABSTRACT

A programmable logic controller (PLC) or programmable controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures. PLCs are used in many industries and machines. Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed or non-volatile memory. A PLC is an example of a real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.

In our everyday cargo or commercial airlines a proper security checkpoint is necessary. The aim of our paper is to create a more systematic and effective checkpoint. The checkpoint will comprise of four stations. The first station will be a conveyor belt that the baggage or cargo is loaded unto; they will then be transported to the following station which is the Testing Station. After completing the Testing Station cargo/luggage will be redirected accordingly. If a scanned object fails the test it will be removed from the line and placed on the platform for further inspection. A Passed pallet will continue after the handling station to another conveyor belt where it will then be removed and packed onto the aircraft in preparation for departure.

Keywords: Security Checkpoint, PLC Automation, Virtual Simulator

1. INTRODUCTION

The successful completion of this automation process will be accompanied by a virtually simulated video that will demonstrate the execution of a written code. To gain the expected results that will make this a more practical system than the systems already in place, we intend to equip our testing station with special gamma and neuron rays that will detect organic and inorganic materials. These rays will then classify the scanned images as either safe or hazardous. The advantages of having a system of this caliber implemented will also be listed and future improvements discussed.

2. Overview of Individual Substations

The three PLC substations (Fig. 1) which will be used to combine as a cargo screening systems are conveyor Belt, Testing and Handling substations. The whole process will start at conveyor belt, which will transport the pallet to the Testing station. The Testing substation will detect any potentially hazardous materials and send the data to next subsystem. Based on data from Testing station, the Handling station will divide the cargo. Failed cargo will be picked up by the Handling station and deposited on platform for further inspection. Passing cargo will continue along to their destination.

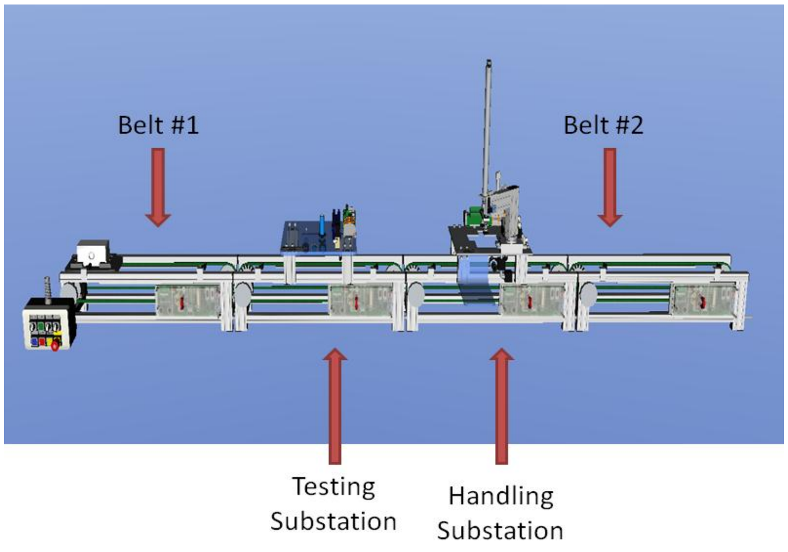


Fig. 1: Automated Cargo Screening Systems

2.1.1 Belt #1

This is the starting point of the carrier, it will begin the process when a pallet is loaded and the start button is pushed. The substation is equipped with sensors that will define its start and end limits.

2.1.2 Real life restrictions to be applied

If there is no carrier and the start button is pressed the process should not begin. A major requirement in a real life application is to always maintain a time efficient and cost effective operation, therefore having the belt run without a carrier in the case where it may have been accidentally pressed should be restricted. A way in which this real life restriction should be implemented is to use an AND gate, this requires a pallet to be detected by a sensor and the start button to be pressed for the process to commence.

Note: Doing this will also provide a safer work environment

2.2.1 Testing Substation

The Testing station will begin when the right most sensor of Belt #1 is activated. When the carrier is detected on the Testing station a stopper will be deployed causing the carrier to stop at a predefined location where the block can be tested. The stopper is controlled by a S_PEXT timer and will run for a specified time, enough for the block to be completely tested. After the time has elapsed the stopper will return to its initial position continuing the carrier forward

2.2.2 Real life restrictions to be applied

When stopper is deployed and testing begins the belt should stop turning. Having the belt turn continuously during testing can be dangerous and as a result cause wear on the belt. This will prove costlier, less productive and a potential hazard.

Executing this process successfully will require us to introduce a S_PEXT timer that will begin when the stopper is deployed. Secondly, when the carrier arrives at stopper and the testing process begins the belt will need to be stopped. This can be done by using the reset logic (R) and is initiated as the testing begins, when the belt on the testing station is stopped we can introduce a NAND that will start a timer which will allow it to restart when the time has elapsed. This will be based on the stopper returning to its initial position and when the testing process is finished.

2.3.1 Handling Station

When the carrier has completely been tested it will proceed to the Handling station. Based on the carriers' results it will either continue through the Handling station (pass) or be removed by the handling station (fail). If a carrier and its contents fail at the Testing station its information will be delivered to the Handling station, a failed carrier will be extracted where it will receive a more detail inspection.

2.3.2 Real life approach

Our system uses pneumatics and a vacuum suction unit that is attached to a 90 degree arm. It will get the job done on a small scale but is not very practical in a real life scenario. In a life model we would want to use more reliable and economical techniques. Replacing the arm with a robotic crane and the pneumatics with hydraulics is one solution to this problem since a cubic volume of oil will do much more work than air. The reason for a robotic crane is to decrease the man power and keep our project consistently efficient.

2.4.1 Belt #2

The final substation of the entire line will be Belt #2 it will be initiated when a carrier that has passed the testing arrives at the right most sensor of the Handling substation. Belt #2 will then continue the carrier to the end of the system where it can then be relocated.

3. Siemens SIMATIC Step 7 Programming

STEP 7 is the basic programming and configuration software from Siemens. It is made up of a series of applications, each of which does a specific job within the scope of programming and automation, such as:

- Configuring and assigning parameters to the hardware
- Creating and debugging user programs
- Configuring networks and connections

The graphic user interface provided for these tasks is known as the SIMATIC Manager (Siemens). The SIMATIC Manager collects all the data and the settings necessary for an automation task together in a project. Within this project the data are structured according to their function and represented as objects.

3.1 LUCAS_NULLE L@Bsoft

To run the program that was created in Sematic Step7, we first have to convert it into the right language format. To do this we will be using Lucas Nulle L@bsoft (LUCAS-NÜLLE). The program which was written using Function Block Diagram must be converted to AWL Instruction List; this is done so the program could be tested on our IMS_Substations.

3.1.1 Programming Languages

Function Block Diagram (FBD) is a graphic representation of the STEP 7 programming language and uses the logic boxes familiar from Boolean algebra to represent the logic. Complex functions (for example, math functions) can be represented directly in conjunction with the logic boxes.

Instruction List language consists of a sequence of instructions, each of which begins on a new line. A line can start with a jump label. This label is followed by an operator and an operand. Operands in instruction lists can comprise variables, literals and names of function blocks. They are combined with operators or functions to form instructions

3.1.2 Bit Logic

The following Bit logics (Norman S. Nise, 2007) were used to write the program in Function Block Diagram language (FBD). Bit logic instructions work with two digits, 1 and 0. These two digits form the base of a number system called the binary system. The two digits 1 and 0 are called binary digits or bits. In the world of contacts and coils, a 1 indicates activated or energized, and a 0 indicates not activated or not energized. The bit logic instructions interpret signal states of 1 and 0 and combine them according to Boolean logic. These combinations produce a result of 1 or 0 that is called the “result of logic operation”.

Logic Gates

- AND Logic Operation
- Assign (=) Operation
- Set Gate (S)
- Reset Output (R)
- Set Reset Flip Flop (SR)
- S_PEXT (Assign Extended Pulse Timer)

3.1.3 AND (&) Logic Operation

The use of an AND logic gate (figure 2) is to check the signal states of two or more specified addresses at the inputs of an AND box. The AND Gate requires two inputs and has one output. The AND Gate only produces an output of 1(true) when both the inputs are a 1, otherwise the output is zero (false). Every AND instruction that is not listed in the beginning of the instruction list of the operation, combines the result of its previous results and the values are stored in the AND truth table (figure 3) based on Boolean algebra A.B

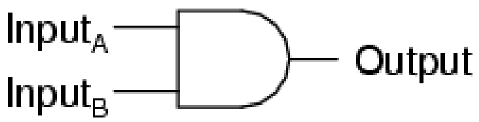


Fig.2: And Gate

Input		Output
A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 3: Truth table

3.1.4 Assign (=) Logic Operation

The Assign gate function is to make the input value equal to the output value. For example when the input is 1(true) the resulting output will also be true.

Boolean algebra A = B

Input	Output
A	B
0	0
1	1

Fig. 4: Truth table

3.1.5 Set (S) Logic Operation

The Set Output instruction is only executed when the result of logic operation is 1 in the address bit, the output will equal 1. If the input = 0, the addressed bit does not change.

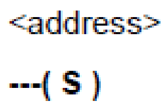


Fig. 5: Set (S) Logic

3.1.6 Set Reset Flip Flop (SR)

SR (Set-Reset Flip Flop) is set if the signal state is "1" at the S input, and "0" at the R input. Otherwise, if the signal state is "0" at the S input and "1" at the R input, the flip flop is reset. The SR flip flop executes first the set instruction then the reset instruction at the specified <address>, so that this address remains reset for the remainder of program scanning.

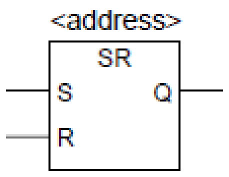


Fig. 6: Flip Flop (SR)

3.1.7 Reset (R) Logic Operation

WHEN THE INPUT IS 1, THE OUTPUT WILL BE ZERO. WHEN THE INPUT IS ZERO THE OUTPUT REMAIN UNCHANGED.

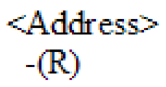


Fig. 7: Description of Reset Logic

3.1.8 S_PEXT (Extended Pulse S5 Timer)

S_PEXT (Extended Pulse S5 Timer) starts the specified timer if there is a positive edge at the start (S) input. A signal change is always necessary in order to enable a timer. The timer runs for the preset time interval specified at input TV even if the signal state at the S input changes to "0" before the time interval has elapsed. The signal state at output Q is "1" as long as the timer is running. The timer will be restarted ("re-triggered") with the preset time value if the signal state at input S changes from "0" to "1" while the timer is running. The timer is reset if the reset (R) input changes from "0" to "1" while the timer is running. The current time and the time base are set to zero.

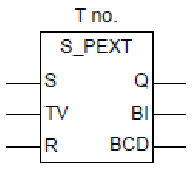
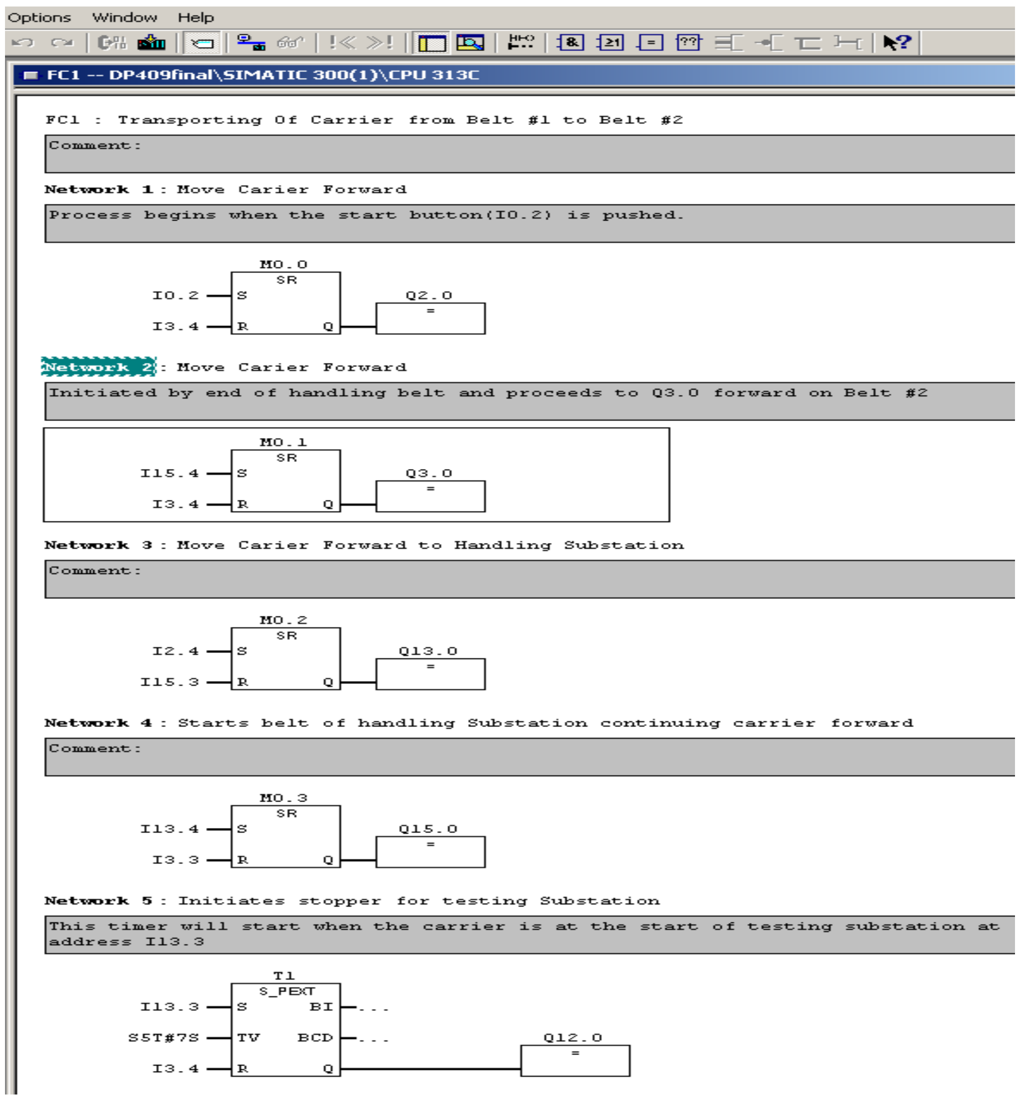


Fig. 8: S_PEXT Timer

4. Snippet of Program used to run Virtual Security Checkpoint (Handling Station)

A PLC program is developed based on bit logic operation to conduct the handling station automation process of security checkpoint (Figure 9). To avoid any programming error, the virtual simulator work bench is used to complete automation process for both sub-systems and system of a security checkpoint.

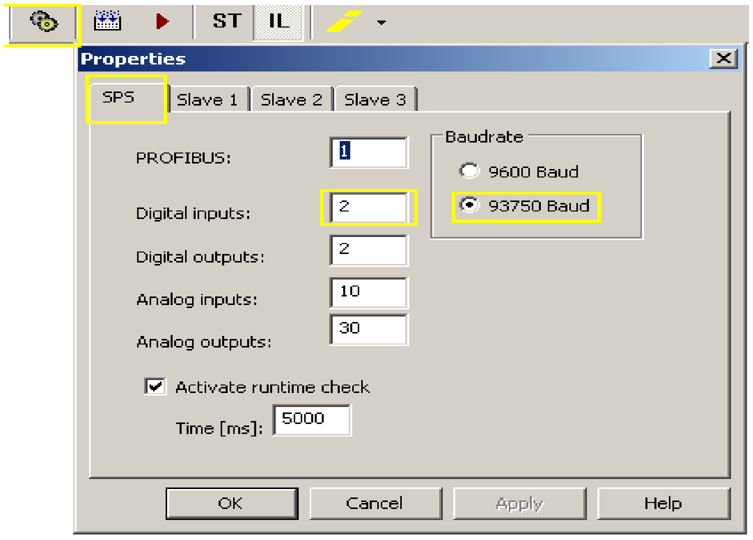


```
FC1 -- DP409final\SIMATIC 300(1)\CPU 313C

FC1 : Transporting Of Carrier from Belt #1 to Belt #2
Comment:

Network 1 : Move Carier Forward
Process begins when the start button(I0.2) is pushed.

            M0.0
             SR
I0.2  ---|S        |     Q2.0
I3.4  ---|R       Q|---[ = ]

Network 2 : Move Carier Forward
Initiated by end of handling belt and proceeds to Q3.0 forward on Belt #2

            M0.1
             SR
I15.4 ---|S        |     Q3.0
I3.4  ---|R       Q|---[ = ]

Network 3 : Move Carier Forward to Handling Substation
Comment:

            M0.2
             SR
I2.4  ---|S        |     Q13.0
I15.3 ---|R       Q|---[ = ]

Network 4 : Starts belt of handling Substation continuing carrier forward
Comment:

            M0.3
             SR
I13.4 ---|S        |     Q15.0
I3.3  ---|R       Q|---[ = ]

Network 5 : Initiates stopper for testing Substation
This timer will start when the carrier is at the start of testing substation at
address I13.3

             T1
           S_PEXT
I13.3  ---|S     BI|-...
S5T#7S ---|TV   BCD|-...              Q12.0
I3.4   ---|R      Q|-----------------[ = ]
```

Fig. 9: Handling station automation programming of security checkpoint

4.1 Conversion of Program from FBD to IL

Our Virtual work bench ensures that costly mistakes are not made while the program is still in its preliminary/development stage making it easy to test the program line by line without any conversion. However to execute the final program that was created in Siemens Step 7 we must convert the Function Block Diagram to Instruction List. The conversion is done by generating sources of the OB1 and FC1. The OB1 is the main callout for the FC1 which contains the Function Block Diagrams. The created sources are then exported to a L@bsoft converter which will then convert and run the program as an Instruction List. Before being able to run the program we must first assign properties to the newly imported program. These properties will consist of digital in/out values, profibus address, master/slave address and finally the communication frequency.



PROFIBUS: 1

Digital inputs: 2

Digital outputs: 2

Analog inputs: 10

Analog outputs: 30

Baudrate: 9600 Baud [✓ 93750 Baud]

[✓] Activate runtime check

Time [ms]: 5000

4.2 Final Version of Executable Program (Snippet)

```
PROGRAM OB1
VAR
LC_bit_0_0 : BOOL;
LC_bit_0_1 : BOOL;
LC_bit_0_2 : BOOL;
LC_bit_0_3 : BOOL;
LC_VKE_BIT : BOOL;
LC_AKKU_1 : DINT;
LC_AKKU_2 : DINT;
LC_timer1_R : BOOL;
LC_timer1_LAST_IN : BOOL;
LC_timer1 : TOF;
LC_timer2_R : BOOL;
LC_timer2_LAST_IN : BOOL;
LC_timer2 : TOF;
LC_timer3_R : BOOL;
LC_timer3_LAST_IN : BOOL;
LC_timer3 : TOF;
END_VAR

VAR CONSTANT
END_VAR

VAR
In_bit_0_2 AT %IX0.2 : BOOL;
In_bit_3_4 AT %IX3.4 : BOOL;
In_bit_15_4 AT %IX15.4 : BOOL;
In_bit_2_4 AT %IX2.4 : BOOL;
In_bit_15_3 AT %IX15.3 : BOOL;
In_bit_13_4 AT %IX13.4 : BOOL;
In_bit_3_3 AT %IX3.3 : BOOL;
In_bit_13_3 AT %IX13.3 : BOOL;
In_bit_1_4 AT %IX1.4 : BOOL;
In_bit_14_4 AT %IX14.4 : BOOL;
In_bit_14_5 AT %IX14.5 : BOOL;
In_bit_14_6 AT %IX14.6 : BOOL;
In_bit_14_3 AT %IX14.3 : BOOL;
END_VAR

VAR
Out_bit_2_0 AT %QX2.0 : BOOL;
Out_bit_3_0 AT %QX3.0 : BOOL;
Out_bit_13_0 AT %QX13.0 : BOOL;
Out_bit_15_0 AT %QX15.0 : BOOL;
Out_bit_12_0 AT %QX12.0 : BOOL;
Out_bit_14_1 AT %QX14.1 : BOOL;
Out_bit_14_0 AT %QX14.0 : BOOL;
Out_bit_14_2 AT %QX14.2 : BOOL;
Out_bit_14_3 AT %QX14.3 : BOOL;
END_VAR
```

5. CONCLUSION

With the ever-present need for technology, great innovations are always welcomed. On our conquest to better command and understand a PLC system we devised a project that would resolve an existing industrial issue. PLC systems are used to reduce the production time and increase the efficiency of an assembly line.

With the right programming background one could easily design a useable PLC program that can be utilized to complete an expected process. The use of virtual software is necessary to execute the program successfully before it is actually distributed to the end user. This initial program that we created is still considered to be in its draft stage, until it has been tested and calibrated on the installation site.

The outline for creating our program was to first identify the pre-existing issue. We then developed a detailed outline of the process. It consisted of everything from the program's initial start condition to the duration of the entire procedure. There are two ways to design a program, the first being trial and error and the second a delineated sketch paired with trial and error. The contrast between the two procedures is the time each takes to complete. Using the independent trial and error process would have been a rigorous and time consuming task.

As engineers we play many roles, a major one of which is Project Managers. Our aim was to consider the efficiency at which the entire project was completed, from design to final execution. Another important role for us as engineers is to troubleshoot our designs before publishing them. This meant making sure our program ran error free continuously for a cycled period. This was to ensure a form of quality control and make certain there were a negligible amount of hidden glitches.

Executing this project from an engineering perspective involved much more than a simple design; it required research and an incorporation of basic design criteria's such as efficiency, reliability and the cost of implementation.

REFERENCES

Nise, M. Norman, (2007). *Control Systems Engineering*, 4th edition, John Wiley.

Siemens Industry, SIMATIC Manager

LUCAS-NÜLLE GmbH, LUCAS-NÜLLE L@Bsoft.

Authorization and Disclaimer