Food Packaging Machine Using PLC

Internship Report

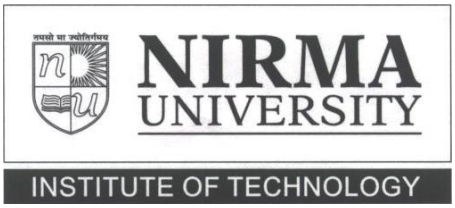
Submitted in Partial Fulfillment of the Requirements
for the Degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & COMMUNICATION ENGINEERING

By
Aman Gupta 13BEC005
Bhuvan Jain 13BEC015
Rajan Kachar 13BEC052
Kshitij Ahuja 13BEC055



Department of Electrical Engineering
Electronics & Communication Engineering Program
Institute of Technology, Nirma University
Ahmedabad-382481
May 2017

CERTIFICATE

This is to certify that the Major Project Report entitled **" Food Packaging Machine Using PLC "** submitted by **Aman Gupta (**Roll No. **13BEC005), Bhuvan Jain (**Roll No. **13BEC015), Rajan Kachar (**Roll No. **13BEC052), Kshitij Ahuja (**Roll No. **13BEC055)** as the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering, Institute of Technology, Nirma University is the record of work carried out by him/her under my supervision and guidance. The work submitted in our opinion has reached a level required for being accepted for the examination.

Date: 17/05/2017

Prof. Bhupendra Ftaniya
Project Guide

Prof. (Dr.) D. K. Kothari
HoD (EC)

Ramdev

SINCE 1965

18th May, 2017

TO WHOM IT MAY CONCERN

This is to certify that **Mr. Aman Gupta** (Roll No. 13BEC005), student of Electronics and Communication Engineering department from Nirma University has undergone his internship with **Ramdev Food Products Pvt. Ltd.,** Bavla, Ahmedabad for the period from 17th January 2017 to 17th May 2017.

During the tenure of his internship, he worked on developing and learning an Application Specific Food Packaging Machine and demonstrated very good skills in PLC programming.

He was diligent and enthusiastic to do his project. He worked with zeal on every challenges faced and possesses good communication skills. He is well organized, autonomous and multitasks to ensure that the project meets its deadlines without any issues.

He completed the project on time and met our expectations to our entire satisfaction.

We wish him best for his career and future endeavours.

Sincerely,

Umesh Khachar

(General Manager- Plant Operations)

RAMDEV FOOD PRODUCTS PVT. LTD.

Regd. Off.: Ramdev Spice World, Changodar - 382 213, Ahmedabad, INDIA.
Ph.: +91 2717 304200 | Fax: +91 2717 304205 | Website: www.ramdev.co.in | E-mail: food@ramdev.co.in
Snacks Plant : Ramdev Food World, Rupal - Chiyada Road, Chiyada - 382 220, Ta.: Bavla,
Dist.: Ahmedabad, INDIA. | Ph.: +91 2714 305500
CIN No.: U15499GJ1989PTC011740

Ramdev

SINCE 1965

18th May, 2017

TO WHOM IT MAY CONCERN

This is to certify that **Mr. Bhuvan Jain** (Roll No. 13BEC015), student of Electronics and Communication Engineering department from Nirma University has undergone his internship with **Ramdev Food Products Pvt. Ltd.,** Bavla, Ahmedabad for the period from 17th January 2017 to 17th May 2017.

During the tenure of his internship, he worked on developing and learning an Application Specific Food Packaging Machine and demonstrated very good skills in PLC programming.

He was diligent and enthusiastic to do his project. He worked with zeal on every challenges faced and possesses good communication skills. He is well organized, autonomous and multitasks to ensure that the project meets its deadlines without any issues.

He completed the project on time and met our expectations to our entire satisfaction.

We wish him best for his career and future endeavours.

Sincerely,

Umesh Khachar

(General Manager- Plant Operations)

RAMDEV FOOD PRODUCTS PVT. LTD.

Regd. Off.: Ramdev Spice World, Changodar - 382 213, Ahmedabad, INDIA.
Ph.: +91 2717 304200 | Fax: +91 2717 304205 | Website: www.ramdev.co.in | E-mail: food@ramdev.co.in
Snacks Plant : Ramdev Food World, Rupal - Chiyada Road, Chiyada - 382 220, Ta.: Bavla,
Dist.: Ahmedabad, INDIA. | Ph.: +91 2714 305500
CIN No.: U15499GJ1989PTC011740

Ramdev

SINCE 1965

18th May, 2017

TO WHOM IT MAY CONCERN

This is to certify that **Mr. Rajan Khachar** (Roll No. 13BEC052), student of Electronics and Communication Engineering department from Nirma University has undergone his internship with **Ramdev Food Products Pvt. Ltd.**, Bavla, Ahmedabad for the period from 17th January 2017 to 17th May 2017.

During the tenure of his internship, he worked on developing and learning an Application Specific Food Packaging Machine and demonstrated very good skills in PLC programming.

He was diligent and enthusiastic to do his project. He worked with zeal on every challenges faced and possesses good communication skills. He is well organized, autonomous and multitasks to ensure that the project meets its deadlines without any issues.

He completed the project on time and met our expectations to our entire satisfaction.

We wish him best for his career and future endeavours.

Sincerely,

Umesh Khachar

(General Manager- Plant Operations)

RAMDEV FOOD PRODUCTS PVT. LTD.

Regd. Off.: Ramdev Spice World, Changodar - 382 213, Ahmedabad, INDIA.
Ph.: +91 2717 304200 | Fax: +91 2717 304205 | Website: www.ramdev.co.in | E-mail: food@ramdev.co.in
Snacks Plant : Ramdev Food World, Rupal - Chiyada Road, Chiyada - 382 220, Ta.: Bavla,
Dist.: Ahmedabad, INDIA. | Ph.: +91 2714 305500
CIN No.: U15499GJ1989PTC011740

Ramdev

SINCE 1965

18th May, 2017

TO WHOM IT MAY CONCERN

This is to certify that **Mr. Kshitij Ahuja** (Roll No. 13BEC055), student of Electronics and Communication Engineering department from Nirma University has undergone his internship with **Ramdev Food Products Pvt. Ltd.,** Bavla, Ahmedabad for the period from 17th January 2017 to 17th May 2017.

During the tenure of his internship, he worked on developing and learning an Application Specific Food Packaging Machine and demonstrated very good skills in PLC programming.

He was diligent and enthusiastic to do his project. He worked with zeal on every challenges faced and possesses good communication skills. He is well organized, autonomous and multitasks to ensure that the project meets its deadlines without any issues.

He completed the project on time and met our expectations to our entire satisfaction.

We wish him best for his career and future endeavours.

Sincerely,

Umesh Khachar

(General Manager- Plant Operations)

RAMDEV FOOD PRODUCTS PVT. LTD.

Regd. Office : Ramdev Spice World, Changodar - 382 213, Ahmedabad, INDIA.
Ph.: +91 2717 304200 | Fax: +91 2717 304205 | Website: www.ramdev.co.in
Snacks Plant : Ramdev Food World, Rupal - Chiyada Road, Chiyada - 382 220, Ta.: Bavla, Dist.: Ahmedabad, INDIA. | E-mail: food@ramdev.co.in
CIN : U15499GJ1989PTC011740

Index

LIST OF FIGURES

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to **Ramdev Food Private Limited** for providing us the opportunity to work under them. We would also like to thank them for helping us understand the various advancements in dedicated PLC systems & understand their implementation in snacks' packaging systems. We would also like to express our whole hearted gratitude to Mr. Umesh Bhai Kachar for guidance & support in developing our skills in PLC applications; for providing us the opportunity to work on the Siemens' S7-200 SIMATIC module & also for his guidance in the programmable logic controls of the PLC system. Without his superior knowledge and experience, the project would lack in quality of outcomes, and thus his support has been instrumental. Also, we sincerely thank him for all the additional knowledge he has imparted to us that has helped us understand the vastness of the technology currently being implemented. Finally, we would like to thank everyone at Ramdev Food Private Limited for their support & patience in bearing with us during the course of our internship.

ABSTRACT

Ramdev Food Pvt. Ltd. Is a private company engaged in providing packaged food items via food packaging sophisticated assembled machinery to wholesale markets. It has state-of-the-art technology that caters to the industry's automation needs. The main technology used as a part of the machinery is Programmable Logic Control (PLC). The interns were introduced to the concept of PLC, SCADA. During the course of the internship, the different coding styles and their applications were acclimatized to the interns. Later on, on different applications, interns were asked to optimize the programs used in industrial machines predominantly for packaging of snacks. The interns were also sensitized as to how SCADA technology worked and its application in fabrication of plastic bags and their dyeing, all the way from plastic seeds. The interns developed PLC based food packaging machine using simple ladder programing involving sensors, stepper motors and their drivers, consequently making them work based on the sensors' stimuli.

Chapter 1

Programmable Logic Controller

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

Before the PLC, control, sequencing, and safety interlock logic for manufacturing automobiles was mainly composed of relays, cam timers, drum sequencers, and dedicated closed-loop controllers. Since these could number in the hundreds or even thousands, the process for updating such facilities for the yearly model change-over was very time consuming and expensive, as electricians needed to individually rewire the relays to change their operational characteristics. The basic function of the S7-200 (a typical PLC and most widely used in India) is to monitor field inputs and, based on your control logic, turn on or off field output devices. This chapter explains the concepts used to execute your program, the various types of memory used, and how that memory is retained.

:

Figure 1: Programmable Logic Controller

Understanding How the S7-200 Executes Your Control Logic

The S7-200 continuously cycles through the control logic in your program, reading and writing data.

The S7-200 Relates Your Program to the Physical Inputs and Outputs

The basic operation of the S7-200 is very simple:
- The S7-200 reads the status of the inputs.
- The program that is stored in the S7-200 uses these inputs to evaluate the control logic. As the program runs, the S7-200 updates the data.
- The S7-200 writes the data to the outputs.

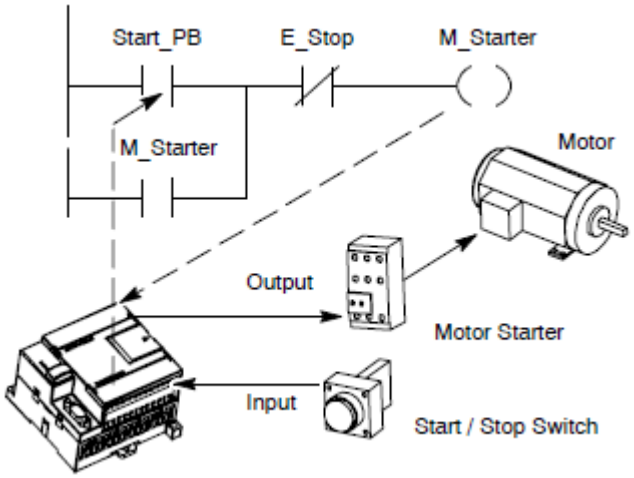


Figure 2 : Electrical Relay diagram of the S7-200.

The calculations of these states then determine the state for the output that goes to the actuator which starts the motor.
Figure 2 shows Controlling Inputs and Outputs

The S7-200 Executes Its Tasks in a Scan Cycle

The S7-200 executes a series of tasks repetitively. This cyclical execution of tasks is called the scan cycle. As shown in Figure 4-2, the S7-200 performs most or all of the following tasks during a scan cycle .Reading the inputs: The S7-200 copies the state of the physical inputs to the process-image input register.

Executing the control logic in the program:

- The S7-200 executes the instructions of the program and stores the values in the various memory areas.

  - Processing any communications requests:

- The S7-200 performs any tasks required for communications.

  - Executing the CPU self-test diagnostics:

- TheS7-200 ensures that the firmware, the program memory, and any expansion modules are working properly.

- Writing to the Outputs:

Writes to the outputs, process any Communications Requests, Perform the CPU Diagnostics

Scan Cycle

Reads the inputs, Execute the Program, Writing to the outputs: The values stored in the process-image output register are written to the physical outputs. The execution of the user program is dependent upon whether the S7-200 is in STOP mode or in RUN mode. In RUN mode, your program is executed; in STOP mode, your program is not executed.

Reading the Inputs

Digital inputs: Each scan cycle begins by reading the current value of the digital inputs and then writing these values to the process-image input register. Analog inputs: The S7-200 does not update analog inputs from expansion modules as part of the normal scan cycle unless filtering of analog inputs is enabled. An analog filter is provided to allow you to have a more stable signal. You can enable the analog filter for each analog input point. When analog input filtering is enabled for an analog input, the S7-200 updates that analog input once per scan cycle, performs the filtering function, and stores the filtered value internally. The filtered value is then supplied each time your program accesses the analog input. When analog filtering is not enabled, the S7-200 reads the value of the analog input from expansion modules each time your program accesses the analog input. Analog inputs AIW0 and AIW2 included on the CPU 224XP are updated every scan with the most recent result from the analog-to-digital converter. This converter is an averaging type (sigma-delta) and those values will usually not need software filtering .

Executing the Program

During the execution phase of the scan cycle, the S7-200 executes your program, starting with the first instruction and proceeding to the end instruction. The immediate I/O instructions give you immediate access to inputs and outputs during the execution of either the program or an interrupt routine. If you use interrupts in your program, the interrupt routines that are associated with the interrupt events are stored as part of the program. The interrupt routines are not executed as part of the
normal scan cycle, but are executed when the interrupt event occurs (which could be at any point in the scan cycle).

Processing Any Communications Requests

During the message-processing phase of the scan cycle, the S7-200 processes any messages that were received from the communications port or intelligent I/O modules.

Executing the CPU Self-test Diagnostics

During this phase of the scan cycle, the S7-200 checks for proper operation of the CPU and forthe status of any expansion modules.

Writing to the Digital Outputs

At the end of every scan cycle, the S7-200 writes the values stored in the process-image output register to the digital outputs. (Analog outputs are updated immediately, independently from the scan cycle.)

Chapter 2

Programming Concepts, Conventions, and Features

The S7-200 continuously executes your program to control a task or process. You use STEP 7-Micro/WIN to create this program and download it to the S7-200. STEP 7--Micro/WIN provides a variety of tools and features for designing, implementing, and debugging your program.

2.1 Guidelines for Designing a Micro PLC System

There are many methods for designing a Micro PLC system. The following general guidelines can apply to many design projects. Of course, you must follow the directives of your own company's procedures and the accepted practices of your own training and location.

2.1.1 Partition Your Process or Machine

Divide your process or machine into sections that have a level of independence from each other. These partitions determine the boundaries between controllers and influence the functional description specifications and the assignment of resources.

2.1.2 Create the Functional Specifications

Write the descriptions of operation for each section of the process or machine. Include the following topics: I/O points, functional description of the operation, states that must be achieved before allowing action for each actuator (such as solenoids, motors, and drives), description of the operator interface, and any interfaces with other sections of the process or machine.

2.1.3 Design the Safety Circuits

Identify equipment requiring hard-wired logic for safety. Control devices can fail in an unsafe manner, producing unexpected startup or change in the operation of machinery. Where unexpected or incorrect operation of the machinery could result in physical injury to people or significant property damage, consideration should be given to the use of electro-mechanical
overrides which operate independently of the S7-200 to prevent unsafe operations. The following tasks should be included in the design of safety circuits:

- Identify improper or unexpected operation of actuators that could be hazardous.
- Identify the conditions that would assure the operation is not hazardous, and determine how to detect these conditions independently of the S7-200.
- Identify how the S7-200 CPU and I/O affect the process when power is applied and removed, and when errors are detected. This information should only be used for designing for the normal and expected abnormal operation, and should not be relied on for safety purposes.
- Design manual or electro-mechanical safety overrides that block the hazardous operation independent of the S7-200.
- Provide appropriate status information from the independent circuits to the S7-200 so that the program and any operator interfaces have necessary information.
- Identify any other safety-related requirements for safe operation of the process.

2.1.4 Specify the Operator Stations

Based on the requirements of the functional specifications, create drawings of the operator stations. Include the following items:

- Overview showing the location of each operator station in relation to the process or machine
- Mechanical layout of the devices, such as display, switches, and lights, for the operator station
- Electrical drawings with the associated I/O of the S7-200 CPU or expansion module

2.1.5 Create the Configuration Drawings

Based on the requirements of the functional specification, create configuration drawings of the control equipment. Include the following items:

- Overview showing the location of each S7-200 in relation to the process or machine
- Mechanical layout of the S7-200 and expansion I/O modules (including cabinets and other equipment)
- Electrical drawings for each S7-200 and expansion I/O module (including the device model numbers, communications addresses, and I/O addresses)

2.1.6 Create a List of Symbolic Names (optional)

If you choose to use symbolic names for addressing, create a list of symbolic names for the absolute addresses. Include not only the physical I/O signals, but also the other elements to be used in your program.

2.1.7 Basic Elements of a Program

A program block is composed of executable code and comments. The executable code consists of a main program and any subroutines or interrupt routines. The code is compiled and downloaded to the S7-200; the program comments are not. You can use the organizational elements (main program, subroutines, and interrupt routines) to structure your control program.

The following example shows a program that includes a subroutine and an interrupt routine. Thissample program uses a timed interrupt for reading the value of an analog input every 100 ms.

Example:Basic Elements of a Program

MAIN	Network 1 SM0.1 SBR_0 EN	Network 1 //On first scan, call subroutine 0. LD SM0.1 CALL SBR_0

SBR0	Network 1
	SM0.0
	MOV_B
	EN ENO
	100 IN OUT SMB34
	ATCH
	EN ENO
	INT_0 INT
	10 EVNT
	(ENI)
		Network 1 //Set the interval to 100 ms
		//for the timed interrupt.
		//Enable interrupt 0.
		LD SM0.0
		MOVB 100, SMB34
		ATCH INT_0, 10
		ENI

INT0	Network 1 SM0.0 MOV_W EN ENO AIW4 IN OUT VW100	Network 1 //Sample the Analog Input 4. LD SM0.0 MOVW AIW4,VW100

Figure 3: Basic Elements of a Program

2.2.1 Subroutines:-

These optional elements of your program are executed only when called: by the main program, by an interrupt routine, or by another subroutine. Subroutines are useful in cases where you want to execute a function repeatedly. Rather than rewriting the logic for each place in the main program where you want the function to occur, you can write the logic once in a subroutine and call the subroutine as many times as needed during the main program. Subroutines provide several benefits:

- Using subroutines reduces the overall size of your program.
- Using subroutines decreases your scan time because you have moved the code out of the main program. The S7-200 evaluates the code in the main program every scan cycle, whether the code is executed or not, but the S7-200 evaluates the code in the subroutine only when you call the subroutine, and does not evaluate the code during the scans in which the subroutine is not called.
- Using subroutines creates code that is portable. You can isolate the code for a function in a subroutine, and then copy that subroutine into other programs with little or no rework.

2.2.2 Interrupt Routines:-

These optional elements of your program react to specific interrupt events. You design an interrupt routine to handle a pre-defined interrupt event. Whenever the specified event occurs, the S7-200 executes the interrupt routine. The interrupt routines are not called by your main program. You associate an interrupt routine with an interrupt event, and the S7-200 executes the instructions in the interrupt routine only on each occurrence of the interrupt event.

3.1 Features:-

3.1.1 Features of the STL Editor

The STL editor displays the program as a text-based language. The STL editor allows you to create control programs by entering the instruction mnemonics. The STL editor also allows you to create programs that you could not otherwise create with the LAD or FBD editors. This is because you are programming in the native language of the S7-200, rather than in a graphical editor where some restrictions must be applied in order to draw the diagrams correctly. As shown in Figure 5-2,
this text-based concept is very similar to assembly language programming.
The S7-200 executes each instruction in the order dictated by the program, from top to bottom, and then restarts at the top. STL uses a logic stack to resolve the control logic. You insert the STL instructions for handling the stack operations.

Sample STL Program
Consider these main points when you select the STL editor:

- STL is most appropriate for experienced programmers.
- STL sometimes allows you to solve problems that you cannot solve very easily with the LAD or FBD editor.
- You can only use the STL editor with the SIMATIC instruction set.
- While you can always use the STL editor to view or edit a program that was created with the LAD or FBD editors, the reverse is not always true. You cannot always use the LAD or FBD editors to display a program that was written with the STL editor.

3.1.2 Features for Debugging the Program

- Bookmarks in your program to make it easy to move back and forth between lines of a long program.

- Cross Reference table allow you to check the references used in your program.
- RUN-mode editing allows you to make small changes to your program with minimal disturbance to the process controlled by the program. You can also download the program block when you are editing in RUN mode.

3.1.3 Features of the LAD Editor

The LAD editor displays the program as a graphical representation similar to electrical wiring diagrams. Ladder programs allow the program to emulate the flow of electric current from a power source through a series of logical input conditions that in turn enable logical output conditions. A LAD program includes a left power rail that is energized. Contacts that are closed allow energy to flow through them to the next element, and contacts that are open block that energy flow.
The logic is separated into networks.The program is executed one network at a time, from left to right and then top to

bottom as dictated by the program. shows an example of a LAD
program. The various instructions are represented by graphic symbols and include three basic forms. Contacts represent logic input conditions such as switches, buttons, or internal conditions.
Coils usually represent logic output results such as lamps, motor starters, interposing relays relays, or internal output conditions. Figure 5-3 Sample LAD Program Boxes represent additional instructions, such as timers, counters, or math instructions.

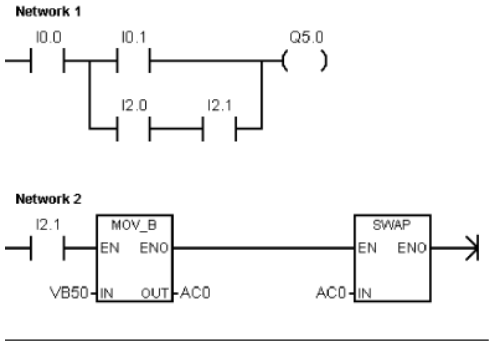


Figure 4: Features of LAD

Consider these main points when you select the LAD editor:
- Ladder logic is easy for beginning programmers to use.
- Graphical representation is easy to understand and is popular around the world.
- The LAD editor can be used with both the SIMATIC and IEC 1131--3 instruction sets.
- You can always use the STL editor to display a program created with the SIMATIC LAD editor.

3.1.4 Features of the FBD Editor

The FBD editor displays the program as a graphical representation that resembles common logic gate diagrams. There are no contacts and coils as found in the LAD editor, but there are equivalent instructions that appear as box instructions. Figure 5-4 shows an example of an FBD program.

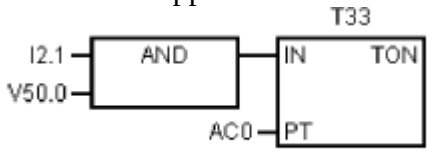


Figure 5: Features of FBD

FBD does not use the concept of left and right power rails; therefore, the term “power flow” is used to express the analogous concept of control flow through the FBD logic blocks. Figure 5-4 Sample FBD Program The logic “1” path through FBD elements is called power flow. The origin of a power flow input and the destination of a power flow output can be assigned directly to an operand. The program logic is derived from the connections between these box instructions. That is, the output from one

instruction (such as an AND box) can be used to enable another instruction (such as a timer) to create the necessary control logic. This connection concept allows you to solve a wide variety of logic problems.
Consider these main points when you select the FBD editor:
- The graphical logic gate style of representation is good for following program flow.
- The FBD editor can be used with both the SIMATIC and IEC 1131--3 instruction sets.
- You can always use the STL editor to display a program created with the SIMATIC FBD editor.

Product Overview:
The S7-200 series of micro-programmable logic controllers (Micro PLCs) can control a wide variety of devices to support your automation needs. The S7-200 monitors inputs and changes outputs as controlled by the user program, which can include Boolean logic, counting, timing, complex math operations, and communications with other intelligent devices. The compact design, flexible configuration, and powerful instruction set combine to make the S7-200 a perfect solution for controlling a wide variety of applications.

What's New?
The new features of the SIMATIC S7-200 include the following. Table 1-1 shows the S7-200 CPUs that support these new features.
- S7-200 CPU models CPU 221, CPU 222, CPU 224, CPU 224XP, and CPU 226 to include: New CPU hardware support: option to turn off run mode edit to get more program memory, CPU 224XP supports onboard analog I/O and two communication ports. CPU 226 includes additional input filters and pulse catch.
- New memory cartridge support: S7-200 Explorer browser utility, memory cartridge transfers, compares, and programming selections
- STEP 7--Micro/WIN, version 4.0, a 32-bit programming software package for the S7-200 to include:
New and improved tools that support the latest CPU enhancements: PID Auto-Tuning Control Panel, PLCs built-in Position Control Wizard, Data Log Wizard, and Recipe Wizard. New diagnostic tool: configuring diagnostic LED New instructions: Daylight Savings time (READ_RTCX and SET_RTCX), Interval Timers (BITIM, CITIM), Clear Interrupt Event (CLR_EVNT), and Diagnostic LED (DIAG_LED). POU and library enhancements: new string constants, added indirect addressing support on more memory types, improved support of the USS library read and write parametrization for Siemens master drives Improved Data Block: Data Block Pages, Data Block auto-increment Improved usability of STEP 7--Micro/WIN

S7-200 CPU:
The S7-200 CPU combines a microprocessor, an integrated power supply, input circuits, and output circuits in a compact housing to create a powerful Micro PLC. See. After you have downloaded your program, the S7-200 contains the logic required to monitor and control the input and output devices in your application.
Siemens provides different S7-200 CPU models with a diversity of features and capabilities that help you create effective solutions for your varied applications. The table below briefly compares some of the features of the CPU.

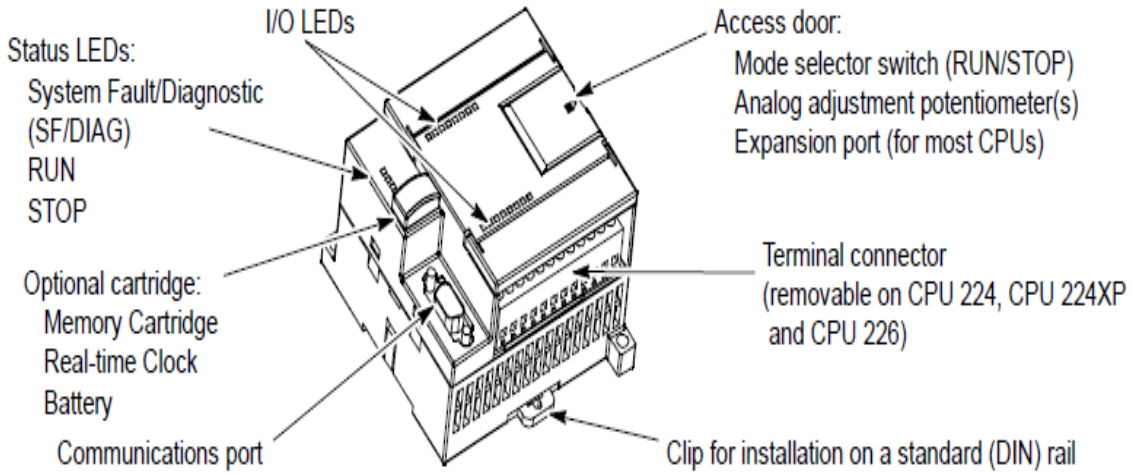


Figure 6: Sophisticated CPU of S7-200

Table 1-2 Comparison of the S7-200 CPU Models

Feature	CPU 221	CPU 222	CPU 224	CPU 224XP	CPU 226
Physical size (mm)	90 x 80 x 62	90 x 80 x 62	120.5 x 80 x 62	140 x 80 x 62	190 x 80 x 62
Program memory: with run mode edit without run mode edit	 4096 bytes 4096 bytes	 4096 bytes 4096 bytes	 8192 bytes 12288 bytes	 12288 bytes 16384 bytes	 16384 bytes 24576 bytes
Data memory	2048 bytes	2048 bytes	8192 bytes	10240 bytes	10240 bytes
Memory backup	50 hours typical	50 hours typical	100 hours typical	100 hours typical	100 hours typical
Local on-board I/O Digital Analog	 6 In/4 Out -	 8 In/6 Out -	 14 In/10 Out -	 14 In/10 Out 2 In/1 Out	 24 In/16 Out -
Expansion modules	0 modules	2 modules[1]	7 modules[1]	7 modules[1]	7 modules[1]
High-speed counters Single phase Two phase	 4 at 30 kHz 2 at 20 kHz	 4 at 30 kHz 2 at 20 kHz	 6 at 30 kHz 4 at 20 kHz	 4 at 30 kHz 2 at 200 kHz 3 at 20 kHz 1 at 100 kHz	 6 at 30 kHz 4 at 20 kHz
Pulse outputs (DC)	2 at 20 kHz	2 at 20 kHz	2 at 20 kHz	2 at 100 kHz	2 at 20 kHz
Analog adjustments	1	1	2	2	2
Real-time clock	Cartridge	Cartridge	Built-in	Built-in	Built-in
Communications ports	1 RS-485	1 RS-485	1 RS-485	2 RS-485	2 RS-485
Floating-point math	Yes				
Digital I/O image size	256 (128 in, 128 out)				
Boolean execution speed	0.22 microseconds/instruction				

S7-200 Expansion Modules:
To better solve your application requirements, the S7-200 family includes a wide variety of expansion modules. You can use these expansion modules to add additional functionality to the S7-200 CPU. The table below provides a list of the expansion modules that are currently available.

Expansion Modules		Types		
Discrete modules	Input	8 x DC In	8 x AC In	16 x DC In
	Output	4 x DC 8 x DC Out	4 x Relays 8 x AC Out	 8 x Relay
	Combination	4 x DC In / 4 x DC Out 4 x DC In / 4 x Relay	8 x DC In / 8 x DC Out 8 x DC In / 8 x Relay	16 x DC In/16 x DC Out 16 x DC In/16 x Relay
Analog modules	Input	4 x Analog In	4 x Thermocouple In	2 x RTD In
	Output	2 x Analog Out		
	Combination	4 x Analog In / 1 Analog Out		
Intelligent modules		Position Ethernet	Modem Internet	PROFIBUS-DP
Other modules		AS-Interface		

Bit Logic Instructions:

Contacts:

Standard Contacts
The Normally Open contact instructions (LD, A, and O) and Normally Closed contact instructions (LDN, AN, ON) obtain the referenced value from the memory or from the process-image register. The standard contact instructions obtain the referenced value from the memory (or process-image register if the data type is I or Q). The Normally Open contact is closed (on) when the bit is equal to 1, and the Normally Closed contact is closed (on) when the bit is equal to 0. In FBD, inputs to both the And and Or boxes can be expanded to a maximum of 32 inputs. In STL, the Normally Open instructions Load, AND, or OR the bit value of the address bit to the top of the stack, and the Normally Closed instructions Load, AND, or OR the logical NOT of the bit value to the top of the stack.

Immediate Contacts
An immediate contact does not rely on the S7-200 scan cycle to update; it updates immediately. The Normally Open Immediate contact instructions (LDI, AI, and OI) and Normally Closed Immediate contact instructions (LDNI, ANI, and ONI) obtain the physical input value when the instruction is executed, but the process-image register is not updated. The Normally Open Immediate contact is closed (on) when the physical input point (bit) is 1, and the Normally Closed Immediate contact is closed (on) when the physical input point (bit) is 0. The Normally Open instructions immediately Load,

AND, or OR the physical input value to the top of the stack, and the Normally Closed instructions immediately Load, AND, or OR the logical NOT of the value of the physical input point to the top of the stack.

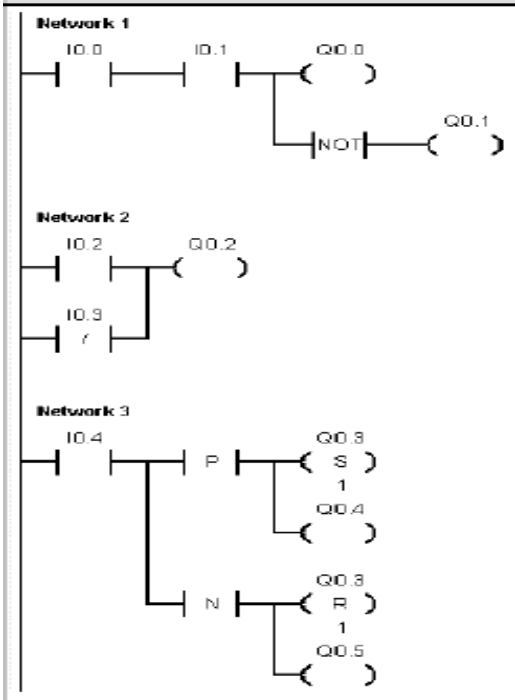
NOT Instruction

The Not instruction (NOT) changes the state of power flow input (that is, it changes the value on the top of the stack from0 to 1 or from1 to 0).

Positive and Negative Transition Instructions

The Positive Transition contact instruction (EU) allows power to flow for one scan for each off-to-on transition. The Negative Transition contact instruction (ED) allows power to flow for one scan for each on-to-off transition. For the Positive Transition instruction, detection of a 0-to-1 transition in the value on the top of the stack sets the top of the stack value to 1; otherwise, it is set
to 0. For a Negative Transition instruction, detection of a 1-to-0 transition in the value on the top of the stack sets the top of the stack value to 1; otherwise, it is set to 0. For run mode editing (when you edit your program in RUN mode), you must enter a parameter for the Positive Transition and Negative Transition instructions. Refer to Chapter 5 for more information about editing in RUN mode.

Example: Contact Instructions




```
Network 1    //N.O. contacts I0.0 AND I0.1 must be on
             //(closed) to activate Q0.0. The NOT
             //instruction acts as an inverter. In RUN
             // mode, Q0.0 and Q0.1 have opposite logic states.
LD     I0.0
A      I0.1
=      Q0.0
NOT
=      Q0.1
Network 2    //N.O. contact I0.2 must be on or N.C.
             //contact I0.3 must be off to activate Q0.2.
             // One or more parallel LAD branches
             //(OR logic inputs) must be true to make
             //the output active.
LD     I0.2
ON     I0.3
=      Q0.2
Network 3     //A positive Edge Up input on a P contact
             //or a negative Edge Down input on a N contact
             //outputs a pulse with a 1 scan cycle
             //duration. In RUN mode, the pulsed state
             //changes of Q0.4 and Q0.5 are too fast to
             // be visible in program status view.
             //The Set and Reset outputs latch the
             // pulse in Q0.3 and make the state
             //change visible in program status view.
LD     I0.4
LPS
EU
S      Q0.3, 1
=      Q0.4
LPP
ED
R      Q0.3, 1
=      Q0.5
```

Figure 7: Contact Instructions

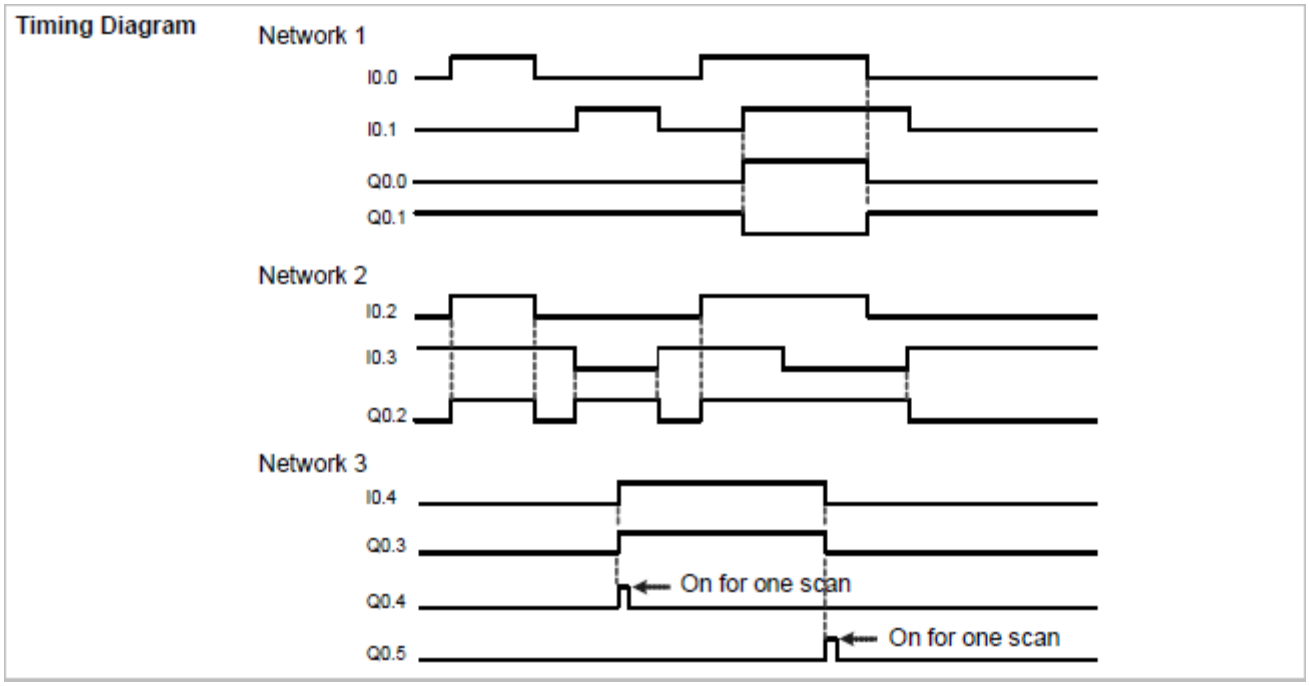


Figure 8: Timing Diagram of S7-200

Logic Stack Instructions

AND Load
The AND Load instruction (ALD) combines the values in the first and second levels of the stack using a logical AND operation. The result is loaded in the top of stack. After the ALD is executed, the stack depth is decreased by one.

OR Load
The OR Load instruction (OLD) combines the values in the first and second levels of the stack, using a logical OR operation. The result is loaded in the top of the stack. After the OLD is executed, the stack depth is decreased by one.

Logic Push
The Logic Push instruction (LPS) duplicates the top value on the stack and pushes this value onto the stack. The bottom of the stack is pushed off and lost.

Logic Read
The Logic Read instruction (LRD) copies the second stack value to the top of stack. The stack is not pushed or popped, but the old top-of-stack value is destroyed by the copy.

Logic Pop
The Logic Pop instruction (LPP) pops one value off of the stack. The second stack value becomes the new top of stack value.

AND ENO

The AND ENO instruction (AENO) performs a logical AND of the ENO bit with the top of the stack to generate the same effect as the ENO bit of a box in LAD or FBD. The result of the AND operation is the new top of stack. ENO is a Boolean output for boxes in LAD and FBD. If a box has power flow at the EN input and is executed without error, the ENO output passes power flow to the next element. You can use the ENO as an enable bit that indicates the successful completion of an instruction. The ENO bit is used with the top of stack to affect power flow for execution of subsequent instructions. STL instructions do not have an EN input. The top of the stack must be a logic 1 for conditional instructions to be executed. In STL there is also no ENO output. However, the STL instructions that correspond to LAD and FBD instructions with ENO outputs set a special ENO bit. This bit is accessible with the AENO instruction.

Load Stack

The Load Stack instruction (LDS) duplicates the stack bit (N) on the stack and places this value on top of the stack. The bottom of the stack is pushed off and lost.

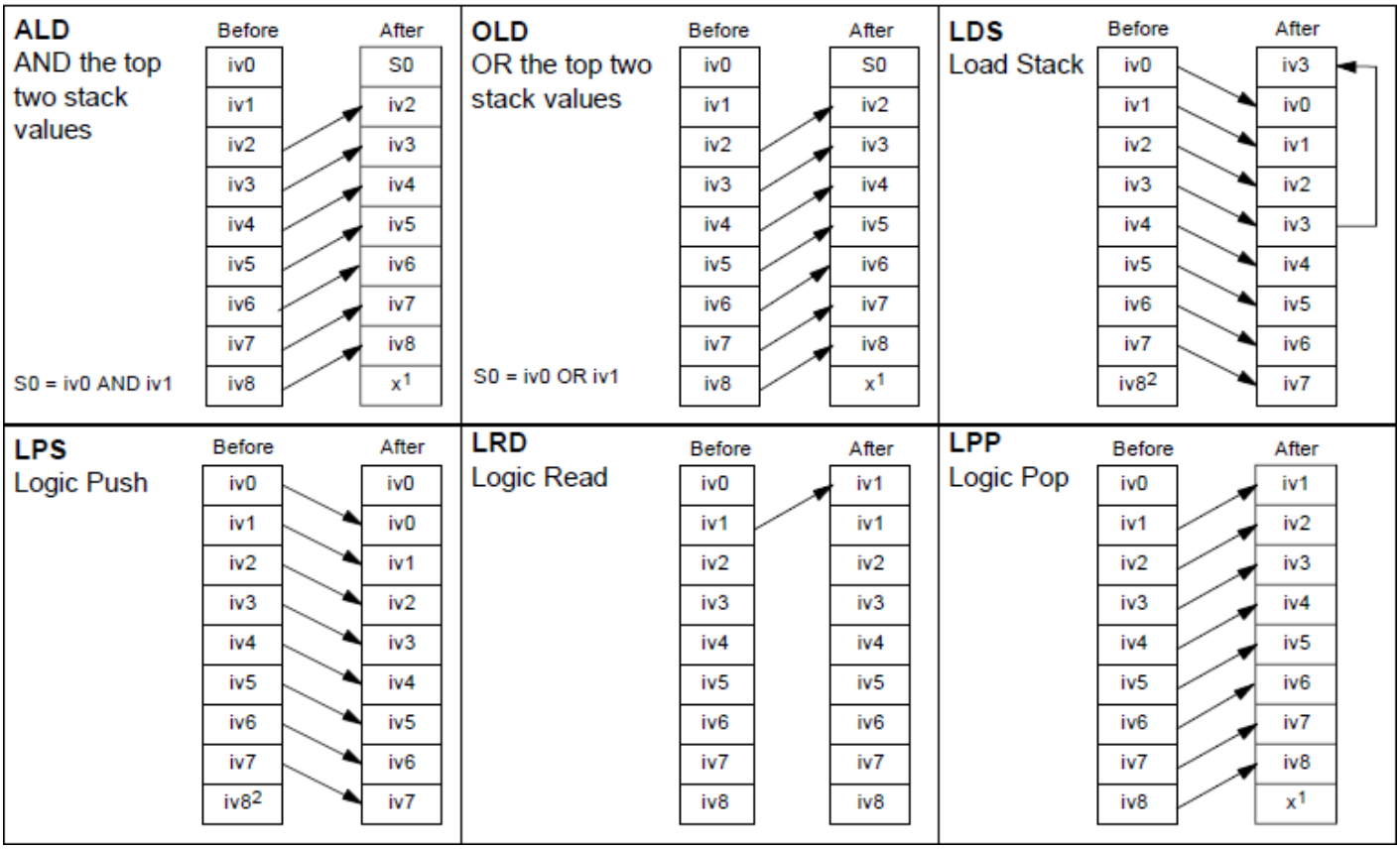


Figure 9: Logical Operations

Set and Reset Dominant Bistable Instructions

The Set Dominant Bistable is a latch where the set dominates. If the set (S1) and reset (R) signals are both true, the output (OUT) is true. The Reset Dominant Bistable is a latch where the reset dominates. If the set (S) and reset (R1) signals are both true, the output (OUT) is false. The Bit parameter specifies the Boolean parameter that is set or reset. The optional output reflects the signal state of the Bit parameter.

Compare Instructions

Comparing Numerical Values

The compare instructions are used to compare two values:
IN1 = IN2 IN1 >= IN2 IN1 <= IN2
IN1 > IN2 IN1 < IN2 IN1 <> IN2
Compare Byte operations are unsigned. Compare Integer operations are signed. Compare Double Word operations are signed.
Compare Real operations are signed. *For LAD and FBD:* When the comparison is true, the Compare instruction turns on the contact (LAD) or output (FBD). *For STL:* When the comparison is true, the Compare instruction Loads, ANDs, or ORs a 1 with the value on the top of the stack (STL). When you use the IEC compare instructions, you can use various data types for the inputs. However, both input values must be of the same data type.

Compare String

The Compare String instruction compares two strings of ASCII characters:
IN1 = IN2 IN1 <> IN2
When the comparison is true, the Compare instruction turns the contact (LAD) or output (FBD) on, or the compare instruction Loads, ANDs or ORs a 1 with the value on the top of the stack (STL).

Conversion Instructions

Standard Conversion Instructions

Numerical Conversions

The Byte to Integer (BTI), Integer to Byte (ITB), Integer to Double Integer (ITD), Double Integer to Integer (DTI), Double Integer to Real (DTR), BCD to Integer (BCDI) and Integer to BCD (IBCD) instructions convert an input value IN to the specified format and stores the output value in the memory location specified by OUT. For example, you can convert a double integer value to a real number. You can also convert between integer and BCD formats.

Round and Truncate

The Round instruction (ROUND) converts a real value IN to a double integer value and places the rounded result into the variable specified by OUT. The Truncate instruction (TRUNC) converts a real number IN into a double integer and places the whole-number portion of the result into the variable specified by OUT.

Segment

The Segment instruction (SEG) allows you to generate a bit pattern that illuminates the segments of a seven-segment display.

Encode and Decode Instructions

Encode

The Encode instruction (ENCO) writes the bit number of the least significant bit set of the input word IN into the least significant “nibble” (4 bits) of the output byte OUT.

Decode
The Decode instruction (DECO) sets the bit in the output word OUT that corresponds to the bit number represented by the least significant "nibble" (4 bits) of the input byte IN. All other bits of the output word are set to 0.

SM Bits and ENO
For both the Encode and Decode instructions, the following conditions affect ENO.

Counter Instructions

SIMATIC Counter Instructions

Count Up Counter
The Count Up instruction (CTU) counts up from the current value each time the count up (CU) input makes the transition from off to on. When the current value Cxx is greater than or equal to the preset value PV, the counter bit Cxx turns on. The counter is reset when the Reset (R) input turns on, or when the Reset instruction is executed. The counter stops counting when it reaches the maximum value (32,767).
STL operation :
H Reset input: Top of stack
H Count Up input: Value loaded in the second stack location

Count Down Counter
The Count Down instruction (CTD) counts down from the current value of that counter each time the count down (CD) input makes the transition from off to on. When the current value Cxx is equal to 0, the counter bit Cxx turns on. The counter resets the counter bit Cxx and loads the current value with the preset value PV when the load input LD turns on. The counter stops upon reaching zero, and the counter bit Cxx turns on.
STL operation:
H Load input: Top of stack
H Count Down input: Value loaded in the second stack location.

Count Up/Down Counter
The Count Up/Down instruction (CTUD) counts up each time the count up (CU) input makes the transition from off to on, and counts down each time the count down (CD) input makes the transition from off to on. The current value Cxx of the counter maintains the current count. The preset value PV is compared to the current value each time the counter instruction is executed. Upon reaching maximum value (32,767), the next rising edge at the count up input causes the current count to wrap around to the minimum value (-32,768). On reaching the minimum value (--32,768), the next rising edge at the countdown input causes the current count to wrap around to the maximum value (32,767). When the current value Cxx is greater than or equal to the preset value PV, the counter bit Cxx turns on. Otherwise, the counter bit turns off. The counter is reset when the Reset (R) input turns on, or when the Reset instruction is executed. The CTUD counter stops counting when it reaches PV.

STL operation:
H Reset input: Top of stack
H Count Down input: Value loaded in the second stack location
H Count Up input: Value loaded in the third stack location

Pulse Output Instruction

The Pulse Output instruction (PLS) is used to control the Pulse Train Output (PTO) and Pulse Width Modulation (PWM) functions available on the high-speed outputs (Q0.0 and Q0.1). The improved Position Control Wizard creates instructions customized to your application that simplify your programming tasks and take advantage of the extra features of the S7-200 CPUs. Refer to Chapter 9 for more information about the Position Control Wizard. You can continue to use the old PLS instruction to create your own motion application, but the linear ramp on the PTO is only supported by instructions created by the improved Position Control Wizard.

PTO provides a square wave (50% duty cycle) output with user control of the cycle time and the number of pulses. PWM provides a continuous, variable duty cycle output with user control of the cycle time and the pulse width. The S7-200 has two PTO/PWM generators that create either a high-speed pulse train or a pulse width modulated waveform. One generator is assigned to digital output point Q0.0, and the other generator is assigned to digital output point Q0.1. A designated special memory (SM) location stores the following data for each generator: a control byte (8-bit value), a pulse count value (an unsigned 32-bit value), and a cycle time and pulse width value (an unsigned 16-bit value). The PTO/PWM generators and the process-image register share the use of Q0.0 and Q0.1. When a PTO or PWM function is active on Q0.0 or Q0.1, the PTO/PWM generator has control of the output, and normal use of the output point is inhibited. The output waveform is not affected by the state of the process-image register, the forced value of the point, or the execution of immediate output instructions. When the PTO/PWM generator is inactive, control of the output reverts to the process-image register. The process-image register determines the initial and final state of the output waveform, causing the waveform to start and end at a high or low level.

Pulse Width Modulation (PWM)

PWM provides a fixed cycle time output with a variable duty cycle. (See Figure 6-30.) You can specify the cycle time and the pulse width in either microsecond or millisecond increments:

Cycle time: 10 μs to 65,535 μs or 2 t 65 535

Pulse width time: 0 μs to 65,535 μs or0 ms to 65,535 ms

There are two different ways to change the characteristics of a PWM waveform: - Synchronous Update: If no time base changes are required, you can use a synchronous update. With a synchronous update, the change in the waveform characteristics occurs on a cycle boundary, providing a smooth transition. - Asynchronous Update: Typically with PWM operation, the pulse width is varied while the cycle time remains constant so time base changes are not required. However, if a change in the time base of the PTO/PWM generator is required, an asynchronous update is used. An asynchronous update causes the PTO/PWM generator to be disabled momentarily, asynchronous to the PWM waveform. This can cause undesirable jitter in the controlled device. For that reason, synchronous PWM updates are recommended. Choose a time base that you expect to work for all of your anticipated cycle time values.

Math Instructions

Add, Subtract, Multiply, and Divide Instructions

Add Subtract

IN1 + IN2 = OUT IN1 -- IN2 = OUT *LAD and FBD* IN1 + OUT = OUT OUT -- IN1 = OUT *STL*

The Add Integer (+I) or Subtract Integer (--I) instructions add or subtract two 16-bit integers to produce a 16-bit result. The Add Double Integer (+D) or Subtract Double Integer (--D) instructions add or

subtract two 32-bit integers to produce a 32-bit result. The Add Real (+R) and Subtract Real (--R) instructions add or subtract two 32-bit real numbers to produce a 32-bit real number result.

Multiply Divide
IN1 * IN2 = OUT IN1 / IN2 = OUT *LAD and FBD* IN1 * OUT = OUT OUT / IN1 = OUT *STL*
The Multiply Integer (*I) or Divide Integer (/I) instructions multiply or divide two 16-bit integers to produce a 16-bit result. (For division, no remainder is kept.) The Multiply Double Integer (*D) or Divide Double Integer (/D) instructions multiply or divide two 32-bit integers to produce a 32-bit result. (For division, no remainder is kept.) The Multiply Real (*R) or Divide Real (/R) instructions multiply or divide two 32-bit real numbers to produce a 32-bit real number result.

SM Bits and ENO
SM1.1 indicates overflow errors and illegal values. If SM1.1 is set, then the status of SM1.0 and SM1.2 is not valid and the original input operands are not altered. If SM1.1 and SM1.3 are not set, then the math operation has completed with a valid result and SM1.0 and SM1.2 contain valid status. If SM1.3 is set during a divide operation,

Multiply Integer to Double Integer and Divide Integer with Remainder

Multiply Integer to Double Integer
IN1 * IN2 = OUT LAD and FBDIN1 * OUT = OUT STL
The Multiply Integer to Double Integer instruction (MUL) multiplies two 16-bit integers and produces a 32-bit product. In the STL MUL instruction, the least-significant word (16 bits) of the 32-bit OUT is used as one of the factors.
Divide Integer with Remainder
IN1 / IN2 = OUT LAD and FBD OUT / IN1 = OUT STL
The Divide Integer with Remainder instruction (DIV) divides two 16-bit integers and produces a 32-bit result consisting of a 16-bit remainder (the most-significant word) and a 16-bit quotient (the least-significant word). In STL, the least-significant word (16 bits) of the 32-bit OUT is used as the dividend.

SM Bits and ENO
For both of the instructions on this page, Special Memory (SM) bits indicate errors and illegal values. If SM1.3 (divide by zero) is set during a divide operation, then the other math status bits are left unchanged. Otherwise, all supported math status bits contain valid status upon completion of the math operation.

Interrupt Instructions

Enable Interrupt and Disable Interrupt
The Enable Interrupt instruction (ENI) globally enables processing of all attached interrupt events. The Disable Interrupt instruction (DISI) globally disables processing of all interrupt events. When you make the transition to RUN mode, interrupts are initially disabled. In RUN mode, you can enable interrupt processing by executing the Enable Interrupt instruction. Executing the Disable Interrupt instruction inhibits the processing of interrupts; however, active interrupt events will continue to be queued. **Error conditions that set ENO = 0:** H 0004 (attempted execution of ENI, DISI, or HDEF instructions in an interrupt routine)

Conditional Return from Interrupt

The Conditional Return from Interrupt instruction (CRETI)can be used to return from an interrupt, based upon thecondition of the preceding logic.

Attach Interrupt

The Attach Interrupt instruction (ATCH) associates an interrupt event EVNT with an interrupt routine number INT and enables the interrupt event. **Error conditions that set ENO = 0:** H 0002 (conflicting assignment of inputs to an HSC)

Detach Interrupt

The Detach Interrupt instruction (DTCH) disassociates an interrupt event EVNT from all interrupt routines and disables the interrupt event.

Clear Interrupt Event

The Clear Interrupt Event instruction removes all interrupt events of type EVNT from the interrupt queue. Use this instruction to clear the interrupt queue of unwanted interrupt events. If this instruction is being used to clear out spurious interrupt events, you should detach the event before clearing the events from the queue. Otherwise new events will be added to the queue after the clear event instruction has been executed. The example shows a high-speed counter in quadrature mode using the CLR_EVNT instruction to remove interrupts. If a light chopper stepper sensor was stopped in a position that is on the edge of a light to dark transition, then small machine vibrations could generate unwanted interrupts before the new PV can be loaded.

Chapter 3

Motor Control With S7-200

Open Loop Motion Control with the S7-200

The S7-200 provides three methods of open loop motion control:

- Pulse Width Modulation (PWM) -- built into the S7-200 for speed, position or duty cycle control
- Pulse Train Output (PTO) -- built into the S7-200 for speed and position control
- EM 253 Position Module -- an add on module for speed and position control

To simplify the use of position control in your application, STEP 7--Micro/WIN provides a Position Control wizard that allows you to completely configure the PWM, PTO or Position module in minutes. The wizard generates position instructions that you can use to provide dynamic control of speed and position in your application. For the Position module STEP 7--Micro/WIN also provides a control panel that allows you to control, monitor and test your motion operations. The S7-200 provides two digital outputs (Q0.0 and Q0.1) that can be configured using the Position Control Wizard for use as either PWM or a PTO outputs. The Position Control Wizard can also be used to configure the EM 253 Position Module.

When an output is configured for PWM operation, the cycle time of the output is fixed and the pulse width or duty cycle of the pulse is controlled by your program. The variations in pulse width can be used to control the speed or position in your application. When an output is configured for PTO operation, a 50% duty cycle pulse train is generated for open loop control of the speed and position for either stepper motors or servo motors. The built in PTO function only provides the pulse train output. Direction and limit controls must be supplied by your application program using I/O built into the PLC or provided by expansion modules. The EM 253 Position Module provides a single pulse train output with integrated direction control, disable and clear outputs. It also includes dedicated inputs which allow the module to be configured for several modes of operation including automatic reference point seek. The module provides a unified solution for open loop control of the speed and position for either stepper motors or servo motors.

To simplify the use of position control in your application, STEP 7--Micro/WIN provides a Position Control wizard that allows you to completely configure the PWM, PTO or Position module in minutes. The wizard generates position instructions that you can use to provide dynamic control of speed and position in your application. For the Position module STEP 7--Micro/WIN also provides a control panel that allows you to control, monitor and test your motion operations. Position Control

3.1 Wizard menu command

1. Select the option to configure the onboard PTO/PWM operation for the S7-200 PLC.
2. Choose the output Q0.0 or Q0.1 that you wish to configure as a PWM output.
3. Next select Pulse Width Modulation (PWM) from the drop down dialog box, select the time base of microseconds or milliseconds and specify the cycle time.

4 Select 4. Finish to complete the wizard. Figure 9-2 Configuring the PWM Output The wizard will generate one instruction for you to use to control the duty cycle of the the PWM output.

3.2 S7-200 Programmable Controller System Manual

The PWMx RUN instruction allows you to control the duty cycle of the output by varying the pulse width from 0 to the pulse width of the cycle time.
The Cycle input is a word value that defines the cycle time for the PWM output. The allowed range is from 2 to 65535 units of the time base (microseconds or milliseconds) that was specified within the wizard. The Duty Cycle input is a word value that defines the pulse width for the PWM output. The allowed range of values is from 0.0 to 65535 units of the time base (microseconds or milliseconds) that was specified within the wizard. The Error is a byte value returned by the PWMx_RUN instruction that indicates the result of execution. See Table for a description of the possible error codes.
Inputs/Outputs Data Types Operands
Cycle, Duty Cycle Word IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant Error Byte IB, QB, VB, MBV, SMB, LB, AC, *VD, *AC, *LD, Constant

3.2.1 Maximum and Start/Stop Speeds

The wizard will prompt you for the maximum speed (MAX_SPEED) and Start/Stop Speed (SS_SPEED) for your application. See Figure 9-3.
- MAX_SPEED: Enter the value for the optimum operating speed of your application within the torque capability of your motor. The torque required to drive the load is determined by friction, inertia, and the acceleration/deceleration times.
- The Position Control wizard calculates and displays the minimum speed that can be controlled by the Position module based on the MAX_SPEED you specify.
- For the PTO output you must specify the desired start/stop speed. Since at least one cycle at the start/stop speed is generated each time a move is executed, use a start/stop speed whose period is less than the acceleration/deceleration time.
- SS_SPEED: Enter a value within the capability of your motor to drive your load at low speeds. If the SS_SPEED value is too low, the motor and load could vibrate or move in short jumps at the beginning and end of travel. If the SS_SPEED value is too high, the motor could lose pulses on start up and the load could overdrive Speed Distance

3.3 Configuring the Motion Profiles

A profile is a pre-defined motion description consisting of one or more speeds of movement that effect a change in position from a starting point to an ending point. You do not have to define a profile in order to use the PTO or the module. The Position Control wizard provides instructions for you to use to control moves without running a profile. A profile is programmed in steps consisting of an acceleration/deceleration to a target speed followed by a fixed number of pulses at the target speed. In the case of single step moves or the last step in a move there is also a deceleration from the target speed (last target speed) to stop. The PTO supports a maximum of 100 profiles, while the module supports a maximum of 25

3.3.1 Profiles.

Defining the Motion Profile
The Position Control wizard guides you through a Motion Profile Definition where you define each motion profile for your application. For each profile, you select the operating mode and define the specifics of each individual step for the profile. The Position Control wizard also allows you to define a symbolic name for each profile by simply entering the symbol name as you define the profile. Selecting the Mode of Operation for the Profile You configure the profile according the the mode of operation desired. The PTO supports relative position and single speed continuous rotation. The Position module

supports absolute position, relative position, single-speed continuous rotation, and two-speed continuous rotation.

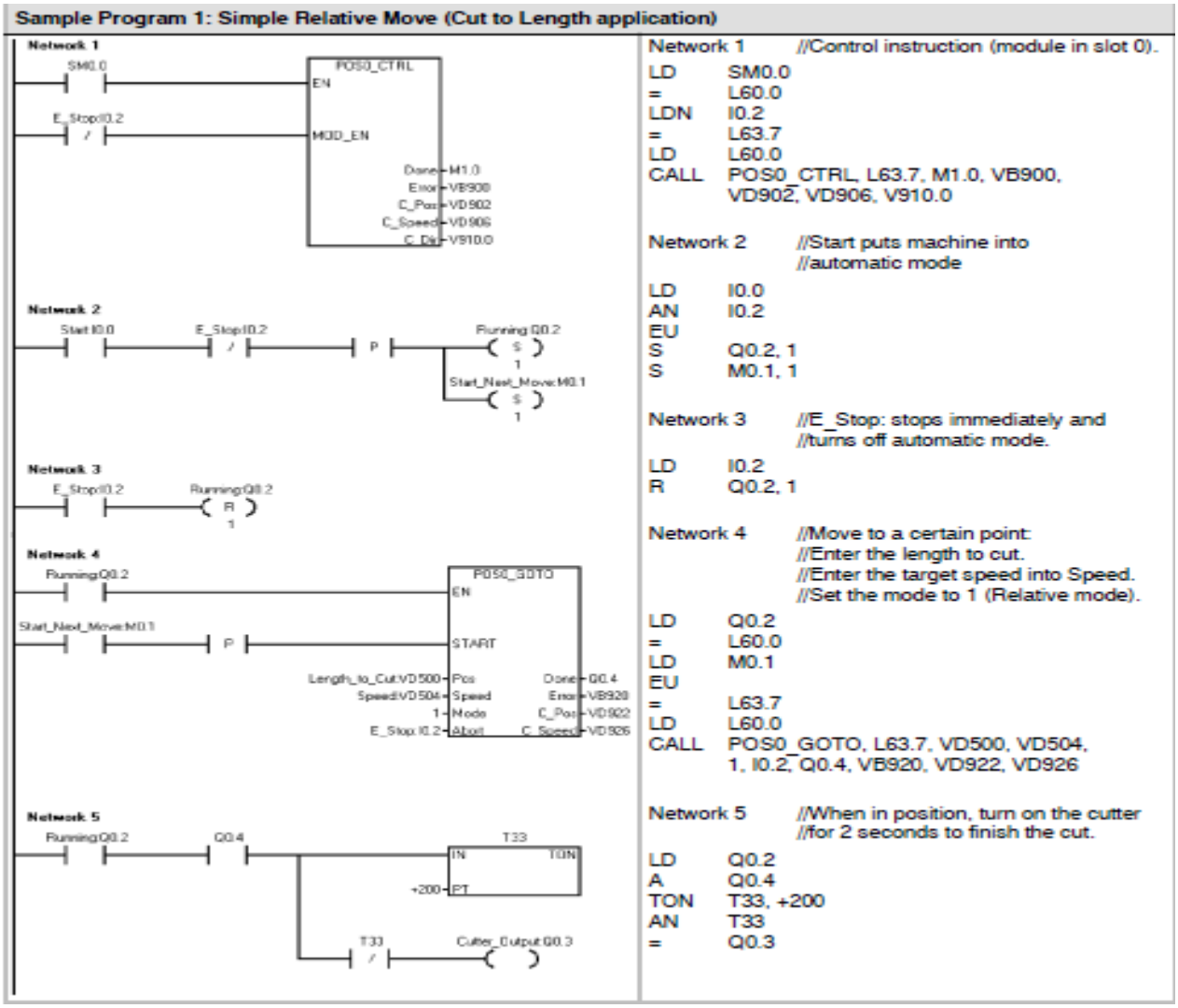


Figure 16: Screenshot of a sample program using Microwin

Creating a Program for the Modem Module

The EM 241 Modem module allows you to connect your S7-200 directly to an analog telephone line, and supports communications between your S7-200 and STEP 7--Micro/WIN. The Modem module also supports the Modbus slave RTU protocol. Communications between the Modem module and the S7-200 are made over the Expansion I/O bus.

3.4.1 Features of the Modem Module

The Modem module allows you to connect your S7-200 directly to an analog telephone line and provides the following features:

- Provides international telephone line interface
- Provides a modem interface to
- Supports the Modbus RTU protocol
- Supports numeric and text paging
- Supports SMS messaging

3.4
3.

- Allows CPU-to-CPU or CPU-to
- Provides password protection
- Provides security callback Country Code Switches
- The Modem module configuration is stored in the CPU .You can use the STEP 7--Micro/WIN Modem Expansion wizard to configure the Modem module.

3.4.2 Sample Program for the Modem Module

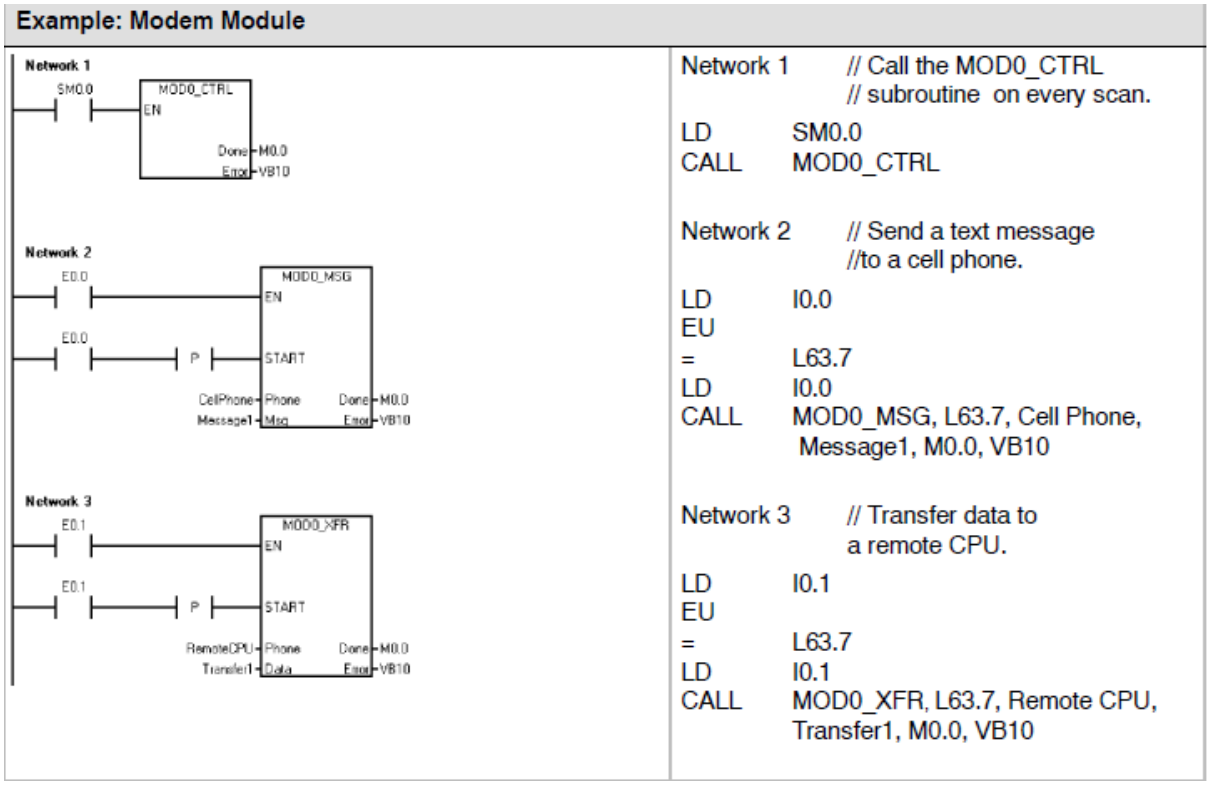


Figure 17: Sample program in the modem module

3.4.3 Special Memory Location for the Modem Module

Fifty bytes of special memory (SM) are allocated to each intelligent module based on its physical position in the I/O expansion bus. When an error condition or a change in status is detected, the module indicates this by updating the SM locations corresponding to the module's position. If it is the first module, it updates SMB200 through SMB249 as needed to report status and error information. If it is the second module, it updates SMB250 through SMB299, and so on.

Chapter 4

Electrical Panel

4.1Switched-Mode Power Supply (SMPS)

A switched-mode power supply (SMPS) is an electronic circuit that converts power using switching devices that are turned on and off at high frequencies, and storage components such as inductors or capacitors to supply power when the switching device is in its non-conduction state.

Switching power supplies have high efficiencies and are widely used in a variety of electronic equipment, including computers and other sensitive equipment requiring stable and efficient power supply.

Figure 18: SMPS block

4.1.1 Topologies of Switch Mode Power Supply

There are different types of topologies for SMPS, among those, a few are as follows

- DC to DC converter
- AC to DC converter
- Fly back converter

1. DC to DC Converter SMPS Working Principle

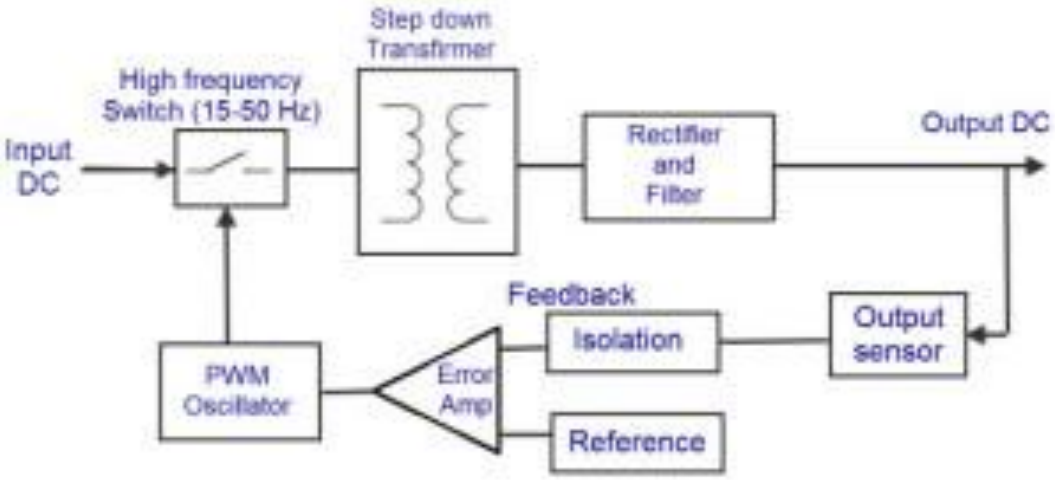


Figure 18: DC to DC convertor SMPS

2. AC to DC Converter SMPS Working Principle

The AC to DC converter SMPS has an AC input. It is converted into DC by rectification process using a rectifier and filter. This unregulated DC voltage is fed to the large-filter capacitor or PFC (Power Factor Correction) circuits for correction of power factor as it is affected. This is because around voltage peaks, the rectifier draws short current pulses having significantly high-frequency energy which affects the power factor to reduce

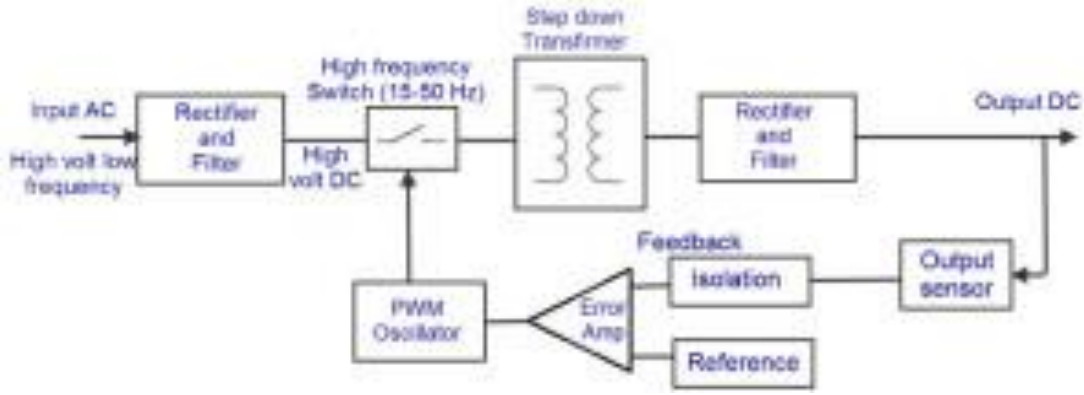


Figure 19: AC to DC convertor SMPS

It is almost similar to the above discussed DC to DC converter, but instead of direct DC power supply, here AC input is used. So, the combination of the rectifier and filter, shown in the block diagram is used for converting the AC into DC and switching is done by using a power MOSFET amplifier with which very high gain can be achieved. The MOSFET transistor has low on-resistance and can withstand high currents. The switching frequency is chosen such that it must be kept inaudible to normal human beings (mostly above 20KHz) and switching action is controlled by a feedback utilizing the PWM oscillator.

This AC voltage is again fed to the output transformer shown in the figure to step down or step up the voltage levels. Then, the output of this transformer is rectified and smoothed by using the output rectifier and filter. A feedback circuit is used to control the output voltage by comparing it with the reference voltage.

3. Fly-back Converter type SMPS Working Principle

The SMPS circuit with very low output power of less than 100W (watts) is usually of Fly-back converter type SMPS, and it is very simple and low- cost circuit compared to other SMPS circuits. Hence, it is frequently used for low-power applications.

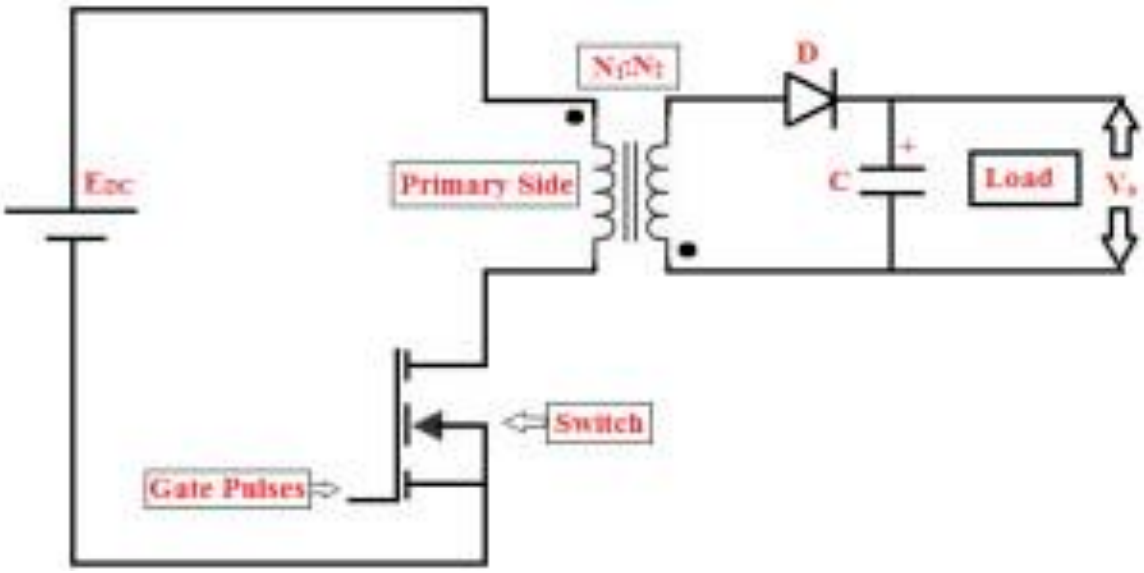


Figure 20: *Fly-back Converter type SMPS*

The unregulated input voltage with a constant magnitude is converted into a desired output voltage by fast switching using a MOSFET; the switching frequency is around 100 kHz. The isolation of voltage can be achieved by using a transformer. The switch operation can be controlled by using a PWM control while implementing a practical fly-back converter.
The two windings of the fly-back transformer act as magnetically coupled inductors. The output of this transformer is passed through a diode and a capacitor for rectification and filtering.

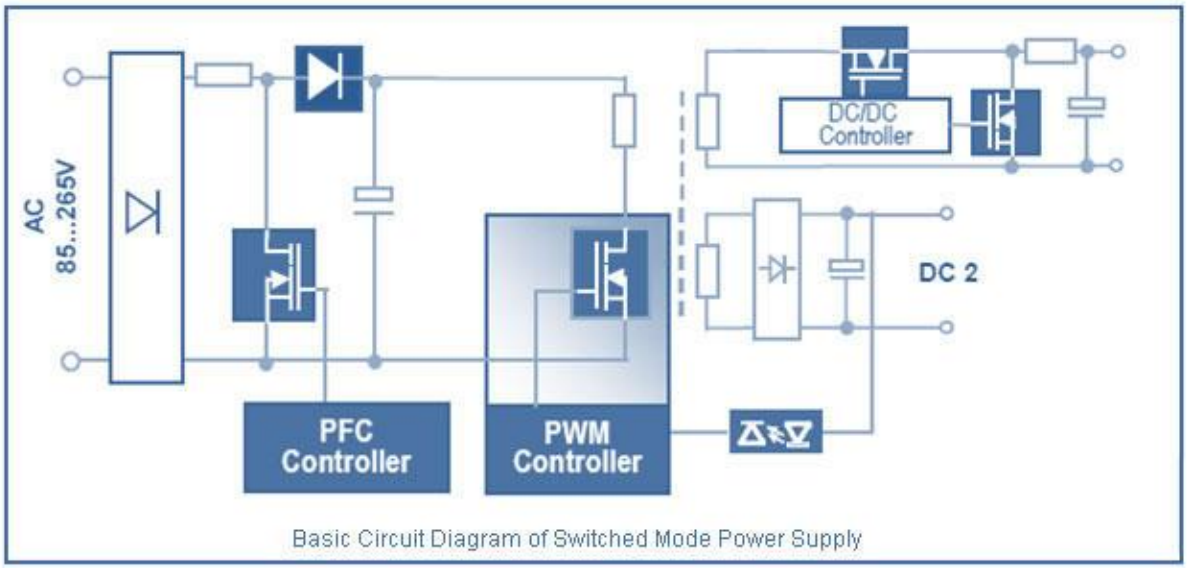
ADVANTAGE

- The switching action means the series regulator element is either on or off. Very high efficiency levels are achieved as very little energy us dissipated as heat.
- As a result of the high efficiency and low levels of heat dissipation, the switch mode power supplies can be compact.
- Switch mode power supply technology also provide high efficiency voltage conversions in voltage step up or “Boost” applications and step down or “Buck” applications.

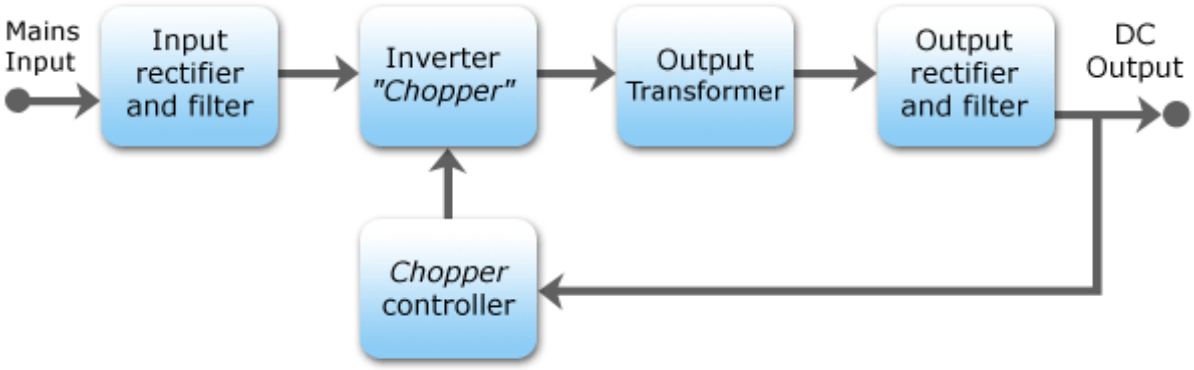
Disadvantage

The transient spikes due to the switching action can migrate into other areas of the circuits if not properly filtered. These can cause electromagnetic or RF interference affecting other nearby items of electronic equipment, particularly if they receive radio signals.

INTERNAL CIRCUIT



Block diagram of a SMPS



1. **Input rectifier stage:**

It is used to convert an ac input to dc. A SMPS with dc input does not require this stage. The rectifier produces unregulated dc which is then passed through the filter circuit.

2. **Inverter stage:**

The inverter stage converts DC, whether directly from the input or from the rectifier stage described above, to AC by running it through a power oscillator, whose output transformer is very small with few windings at a frequency of tens or hundreds of kilohertz.

3. Output transformer:
If the output required is to be isolated from input, the inverted AC is used to draw the primary windings of a high frequency transformer. This converts the voltage up or down to the required output level on it's secondary winding.

4. Output rectifier:
If the dc output is required, the ac output from the transformer is rectified.

Regulation:

Feedback circuit monitors the output voltage and compares it with the reference voltage.

Factors to be considered while selecting a topology for a particular application: -

1. Is input-to-output dielectric isolation required for the application?
2. Are multiple outputs required?
3. Does the prospective topology place a reasonable voltage stress across the voltage semiconductors?
4. Does the prospective topology place a reasonable current stress across the voltage semiconductors?
5. How much of the input voltage is placed across the primary transformer winding or inductor?

Typical maximum output power available from each topology:

Converter Topology	Maximum output power
Flyback	200W
Forward	300W
Push-pull	500W
Half bridge	1000W
Full bridge	>1000W

4.2 WHAT IS PROXIMITY SENSOR

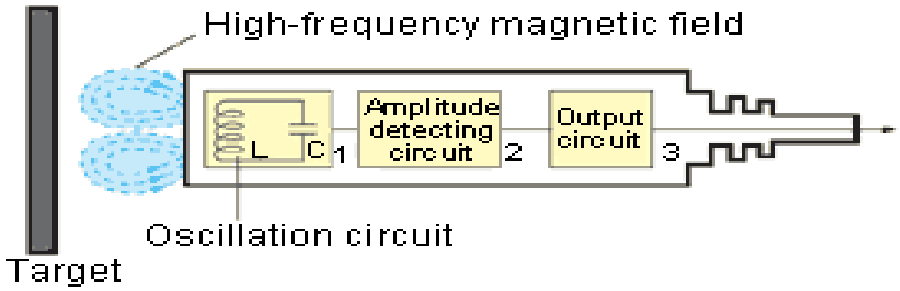
When anything comes in the range of Proximity sensor it flash out infrared beam and monitors reflections. When sensor senses reflections it confirms that there's an object nearby.

Types of proximity sensor

- INDUCTIVE PROXIMITY SENSOR
- CAPACITIVE PROXIMITY SENSOR
- ULTRASONIC PROXIMITY SENSOR
- OPTICAL PROXIMITY SENSOR

INDUCTIVE PROXIMITY SENSOR

An Inductive proximity sensor is an electronic proximity sensor , which detect metallic object without touching them. Their operating principle is based on a coil and high frequency oscillator that creates a field in the close surrounding of the sensing surface, The operating distance of the sensor depends on the coil's size as well as the target 's shape, size and material. The Main Components Of The Inductive Proximity Sensor Are. Coil, Oscillator, Detector And The Output Circuit .The Operating Distance Of The Sensor Depends On The Actuator's Shape And Size And Is Strictly Linked To The Nature Of The Material. The Coil Generates The High Frequency Magnetic Field In Front Of The Face. When The Metallic Target Comes In This Magnetic Field It Absorbs Some Of The Energy. Hence the oscillator field is affected. The rise or fall of such oscillation is identified by a threshold circuit. that changes the output of the sensor.



ADVANTAGES

- they are very accurate compared to other technologies.
- Have high switching rate.
- Can work in harsh environmental conditions.

DISADVANTAGES

- It can detect only metallic target.
- Operating range may be limited.

Capacitive Proximity Sensors

It can also detect metals but along with it can also detect resins, liquids, powders

Magnetic Proximity Sensors

Name is saying its sensing object – magnets. Magnetic Proximity Sensors have no electrical noise effect and it can work on DC, AC, AC/DC, DC. Again sensing distance can vary due to factors such as the temperature, the sensing object, surrounding objects, and the mounting distance between Sensors. This type of sensors have highest sensing range upto 120 mm.

- But in this machine we use inductive proximity sensor

Example of PLC coding : A sample mixing tank.

Step 1: Fill the tank with Ingredient 1.

Step 2: Fill the tank with Ingredient 2.

Step 3: Monitor the tank level for closure of the high-level switch.

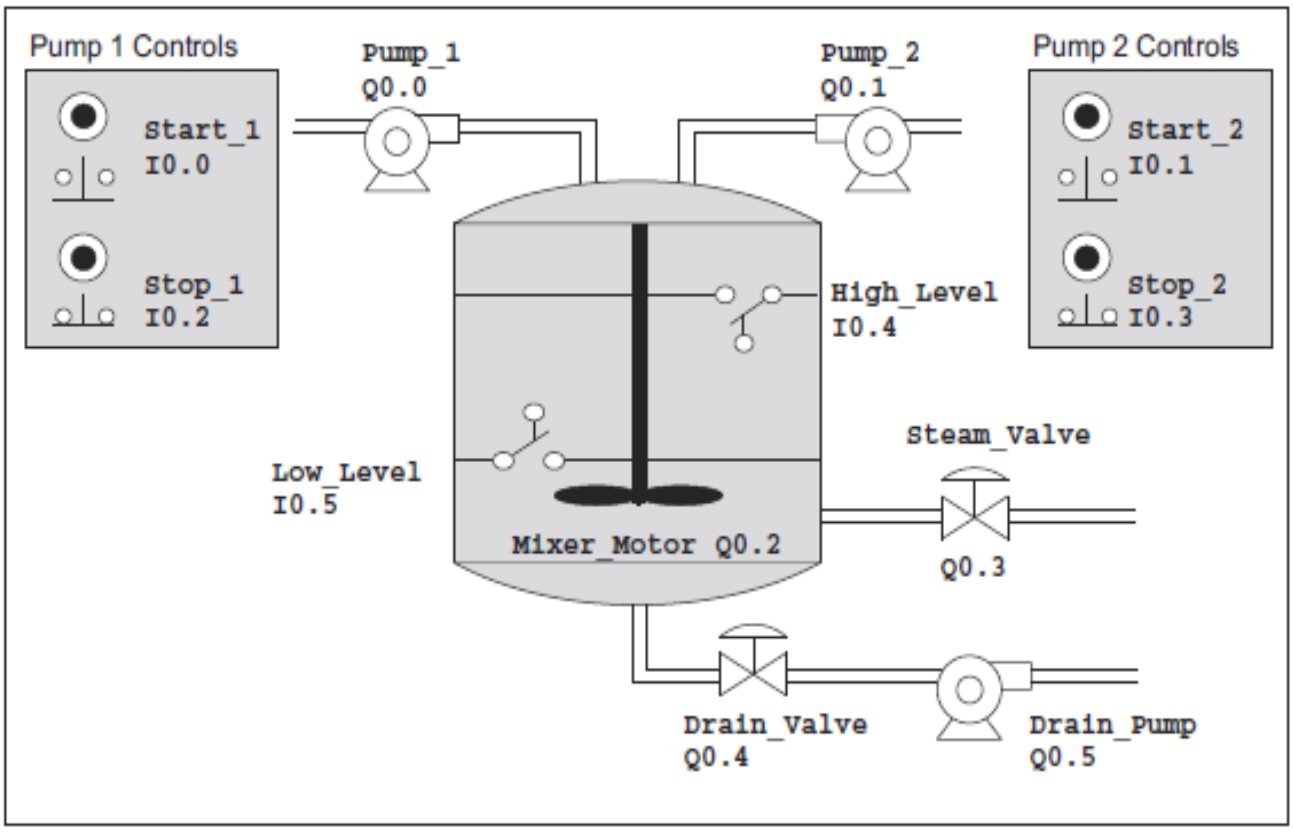
Step 4: Maintain the pump status if the start switch opens.

Step 5: Start the mix-and-heat cycle.

Step 6: Turn on the mixer motor and steam valve.

Step 7: Drain the mixing tank.

Step 8: Count each cycle.



STL Coding

```
STL                              Description
NETWORK 1                        //Pill the tank with Ingredient 1.
LD      "Start_1"
O       "Pump_1"
A       "Stop_1"
AN      "High_Level"
=       "Pump_1"

NETWORK 2                        //Pill the tank with Ingredient 2.
LD      "Start_2"
O       "Pump_2"
A       "Stop_2"
AN      "High_Level"
=       "Pump_2"

NETWORK 3                        //Set memory bit if high level is reached.
LD      "High_Level"
S       "Hi_Lev_Reached", 1

NETWORK 4                        //Start timer if high level is reached.
LD      "Hi_Lev_Reached"
TON     "Mix_Timer", +100

NETWORK 5                        //Turn on the mixer motor.
LDN     "Mix_Timer"
A       "Hi_Lev_Reached"
=       "Mixer_Motor"
=       "Steam_Valve"

NETWORK 6                        //Drain the mixing tank.
LD      "Mix_Timer"
AN      "Low_Level"
=       "Drain_Valve"
=       "Drain_Pump"

NETWORK 7                        //Count each cycle.
LD      "Low_Level"
A       "Mix_Timer"
LD      "Reset"
CTU     "Cycle_Counter", +12

NETWORK 8                        //Reset memory bit if low level or time-out.
LD      "Low_Level"
A       "Mix_Timer"
R       "Hi_Lev_Reached", 1

NETWORK 9                        //End the main program.
MEND
```

Ladder Diagram

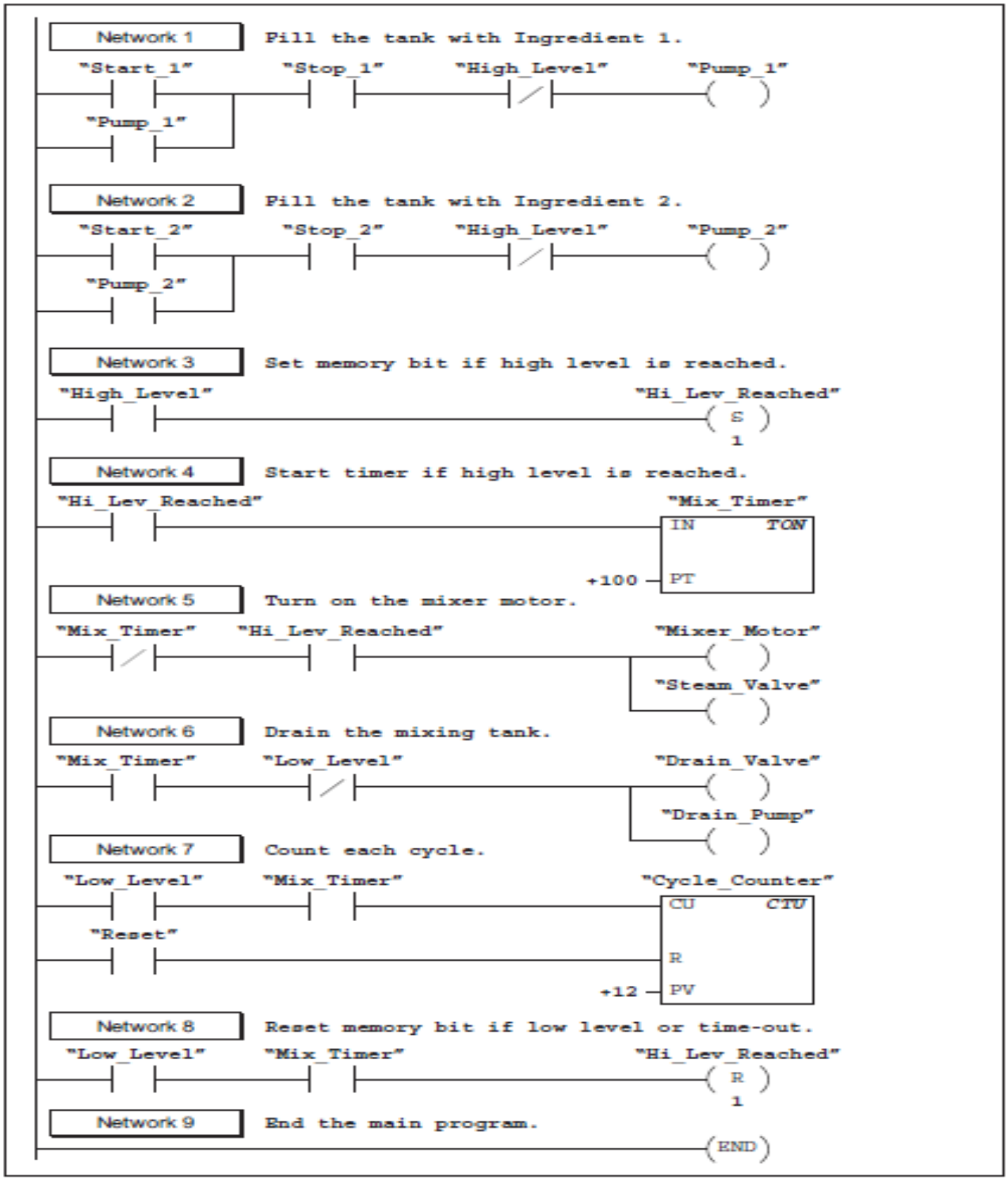


Figure 4-3 Sample Program in Ladder Logic

Symbol List

Symbol Table - c:\microwin\project1.sym

Symbol Name	Address	Comment
Start_1	I0.0	Start switch for Paint Ingredient 1
Start_2	I0.1	Start switch for Paint Ingredient 2
Stop_1	I0.2	Stop switch for Paint Ingredient 1
Stop_2	I0.3	Stop switch for Paint Ingredient 2
High_Level	I0.4	Limit switch for maximum level in tank
Low_Level	I0.5	Limit switch for minimum level in tank
Reset	I0.7	Reset control for counter
Pump_1	Q0.0	Pump for Paint Ingredient 1
Pump_2	Q0.1	Pump for Paint Ingredient 2
Mixer_Motor	Q0.2	Motor for the tank mixer
Steam_Valve	Q0.3	Steam to heat mixture in tank
Drain_Valve	Q0.4	Valve to allow mixture to drain
Drain_Pump	Q0.5	Pump to drain mixture out of tank
Hi_Lev_Reached	M0.1	Memory bit
Mix_Timer	T37	Timer to control mixing/heating mixture
Cycle_Counter	C30	Counts total mix & heat cyles completed

S7 1214C Data Sheet

Supply voltage
Rated value (AC)
- 120 V AC Yes
- 230 V AC Yes

Line frequency
- permissible range, lower limit 47 Hz
- permissible range, upper limit 63 Hz

Input current
Current consumption (rated value) 100 mA at 120 V AC; 50 mA at 240 V AC
Current consumption, max. 300 mA at 120 V AC; 150 mA at 240 V AC

Memory
Work memory
- integrated 100 kbyte
- expandable No

Backup
- present Yes
- maintenance-free Yes
- without battery Yes

CPU processing times
for bit operations, typ. 0.085 µs; / instruction
for word operations, typ. 1.7 µs; / instruction
for floating point arithmetic, typ. 2.3 µs; / instruction

Digital inputs
Number of digital inputs 14

Input voltage
- Rated value (DC) 24 V
- for signal "0" 5 V DC at 1 mA
- for signal "1" 15 V DC at 2.5 mA

Digital outputs
Number of digital outputs 10; Relays

Analog inputs
Number of analog inputs 2

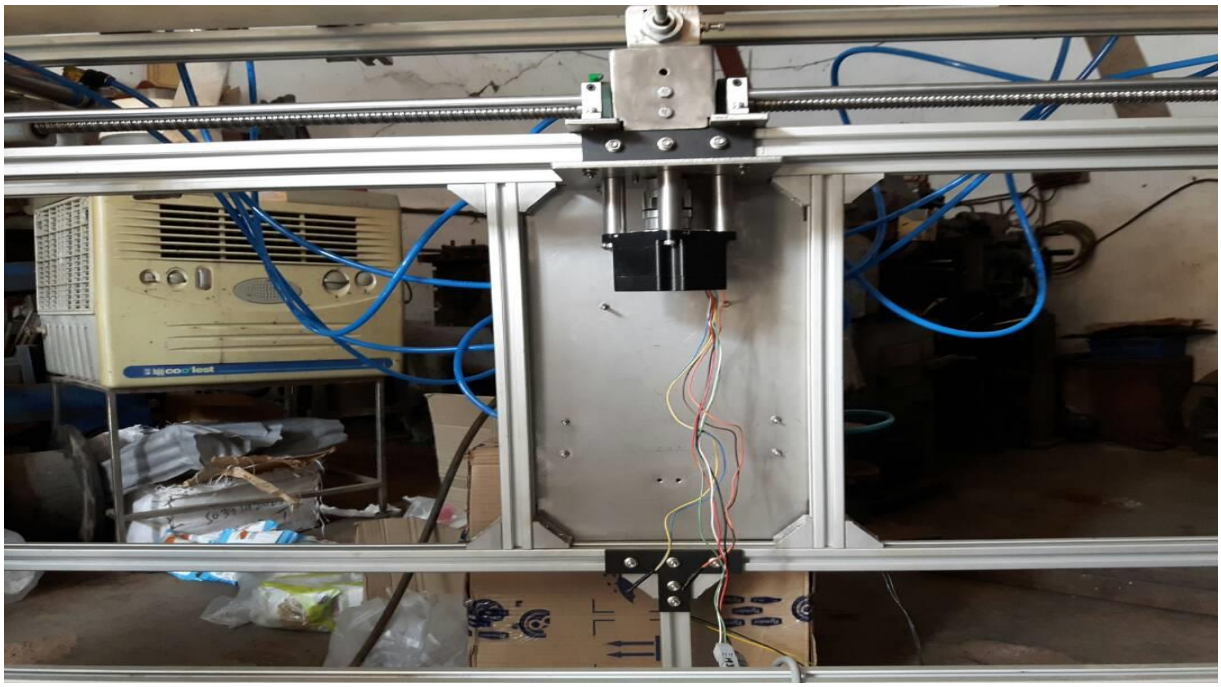
Analog outputs
Number of analog outputs 0

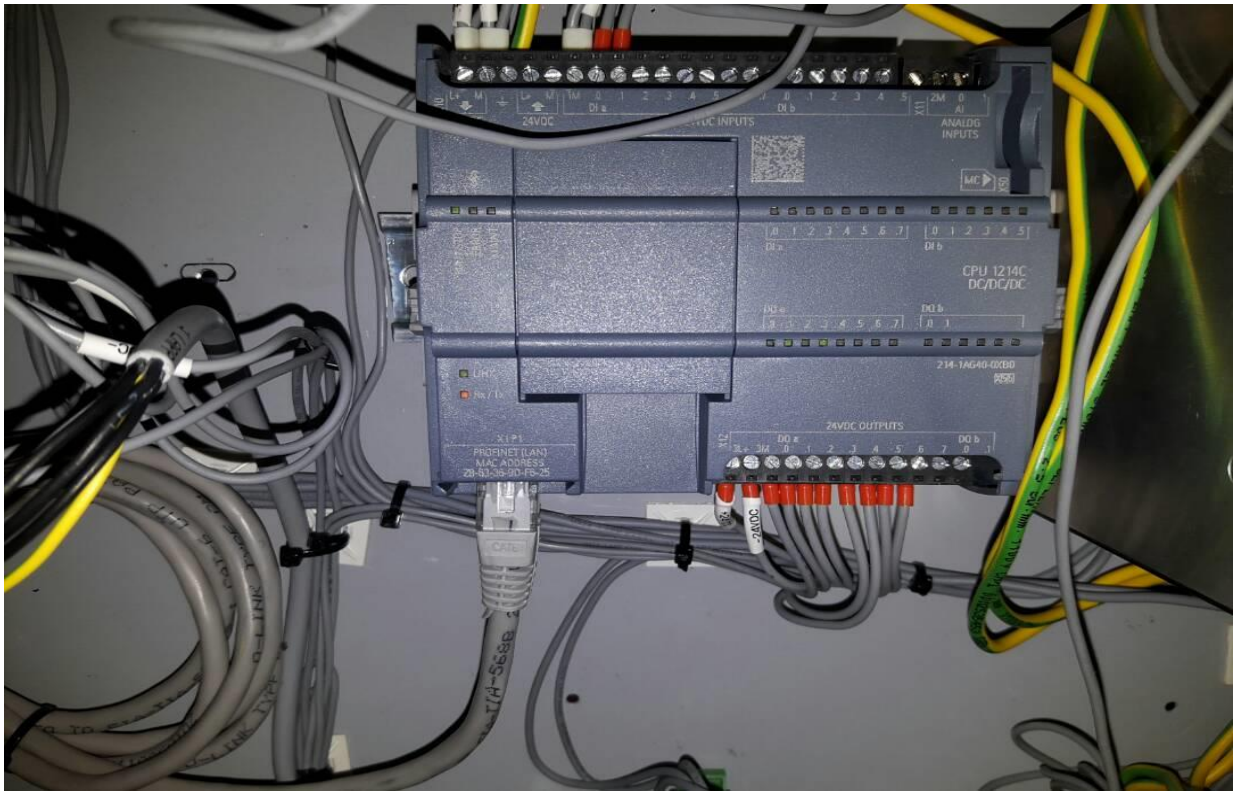
Interface	
Interface type	PROFINET
Physics	Ethernet
Protocols (Ethernet)	
● TCP/IP	Yes
● DHCP	No
● SNMP	Yes
● DCP	Yes
● LLDP	Yes

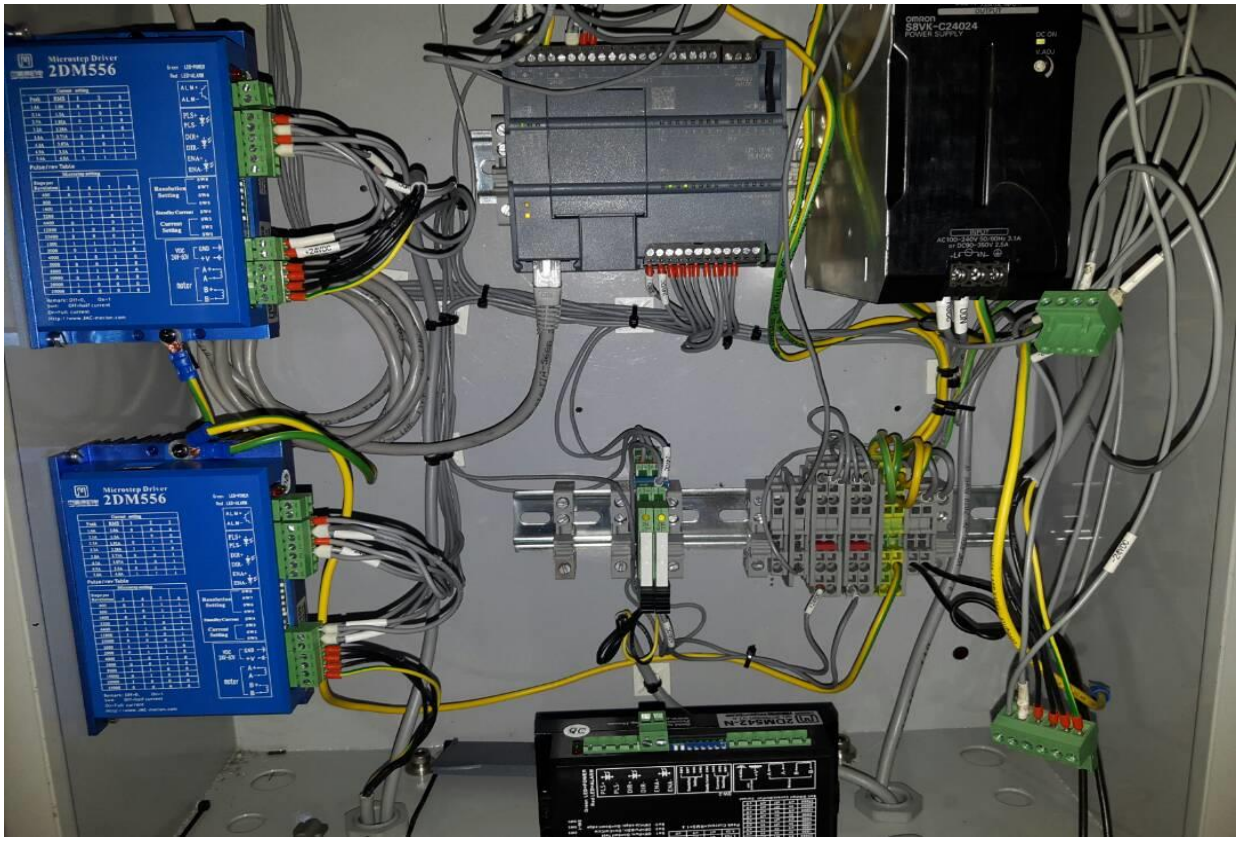
Chapter 5

Results

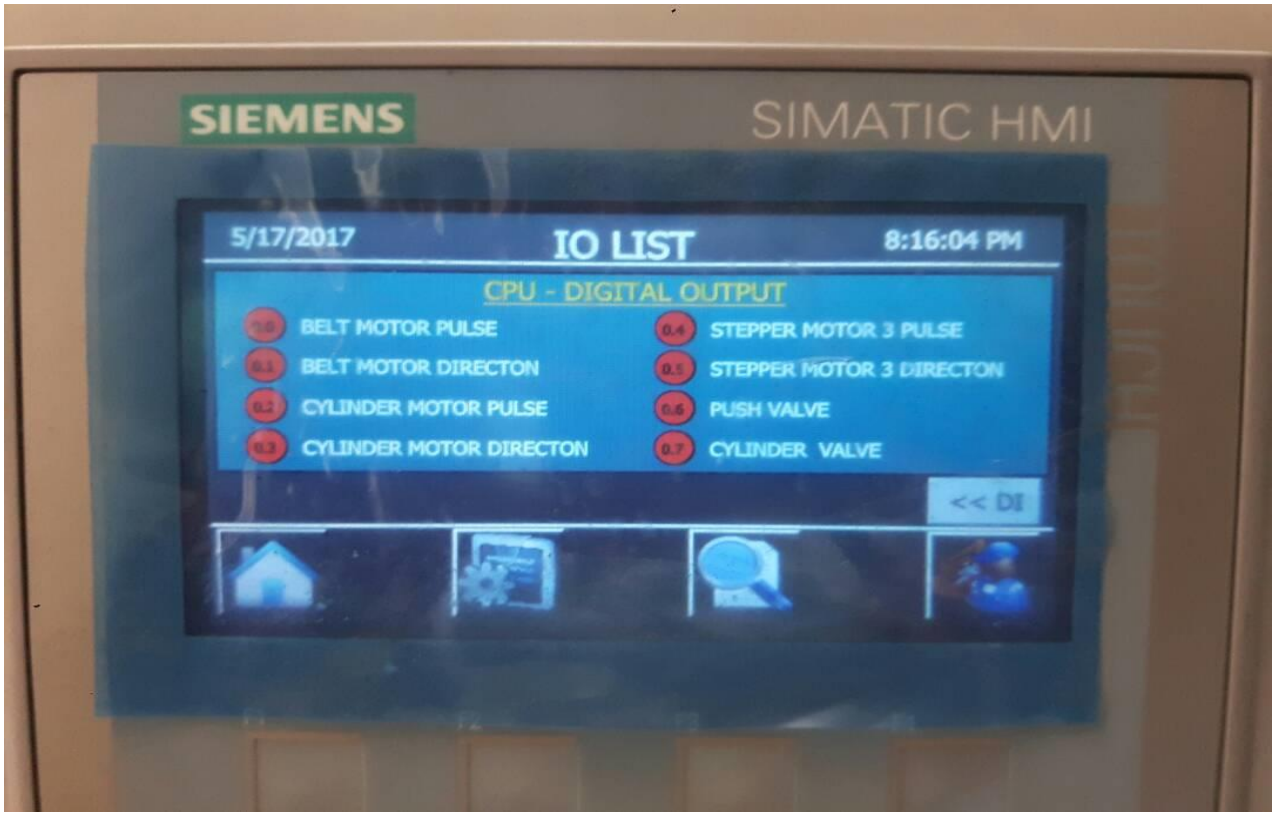
5.1 Final Machine

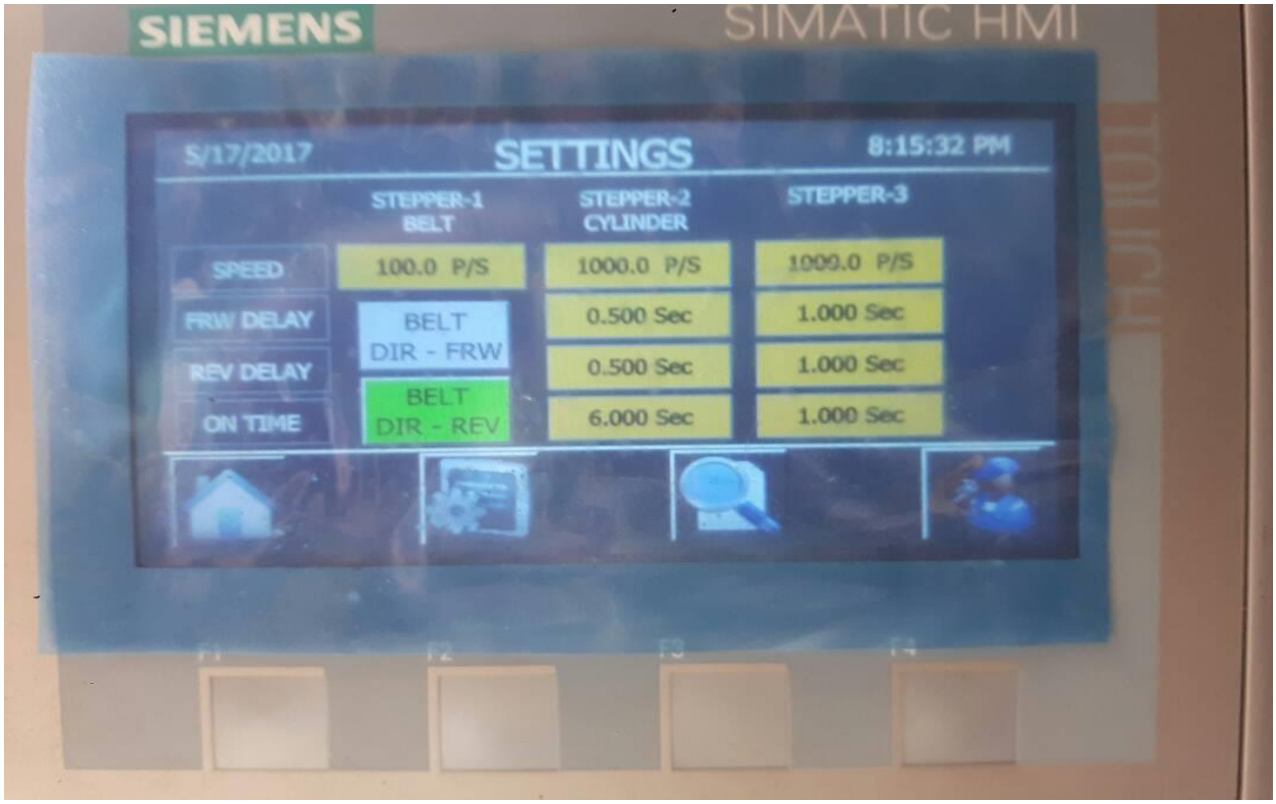


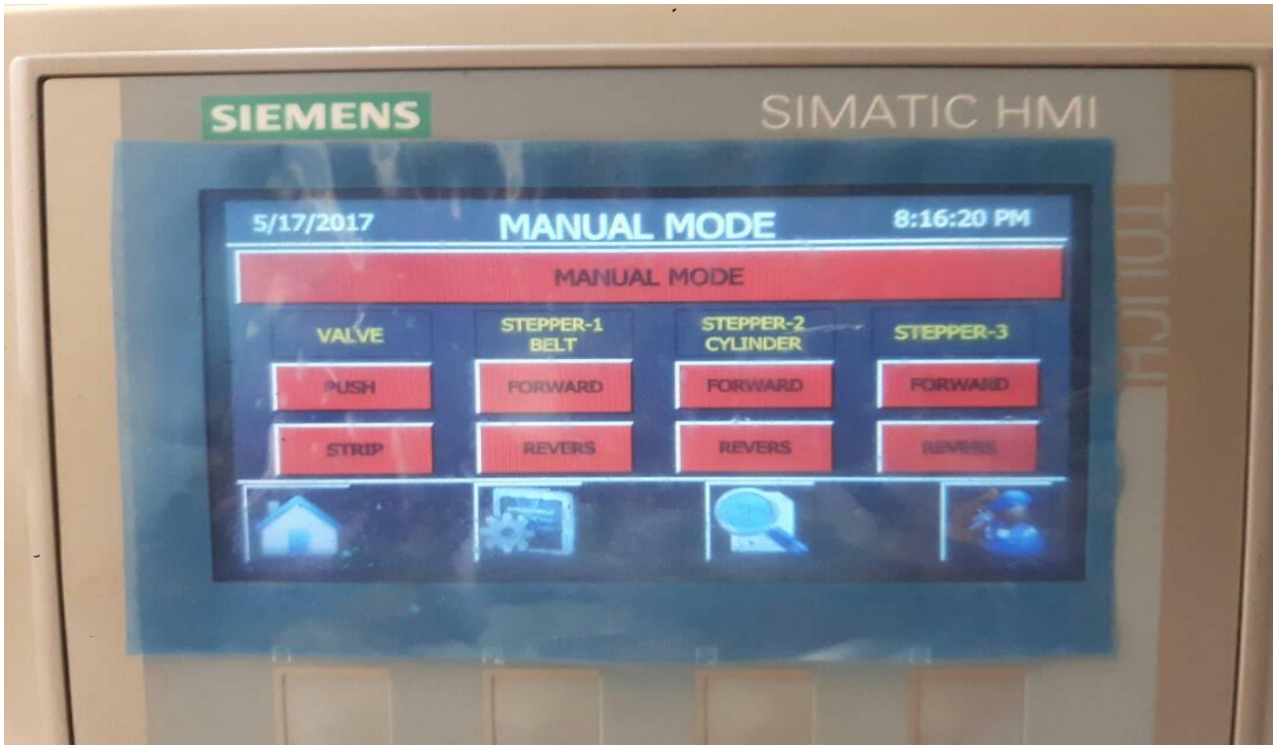

24VDC
DI a
DI b
ANALOG INPUTS
CPU 1214C
DC/DC/DC
24VDC OUTPUTS
PROFINET (LAN)
MAC ADDRESS

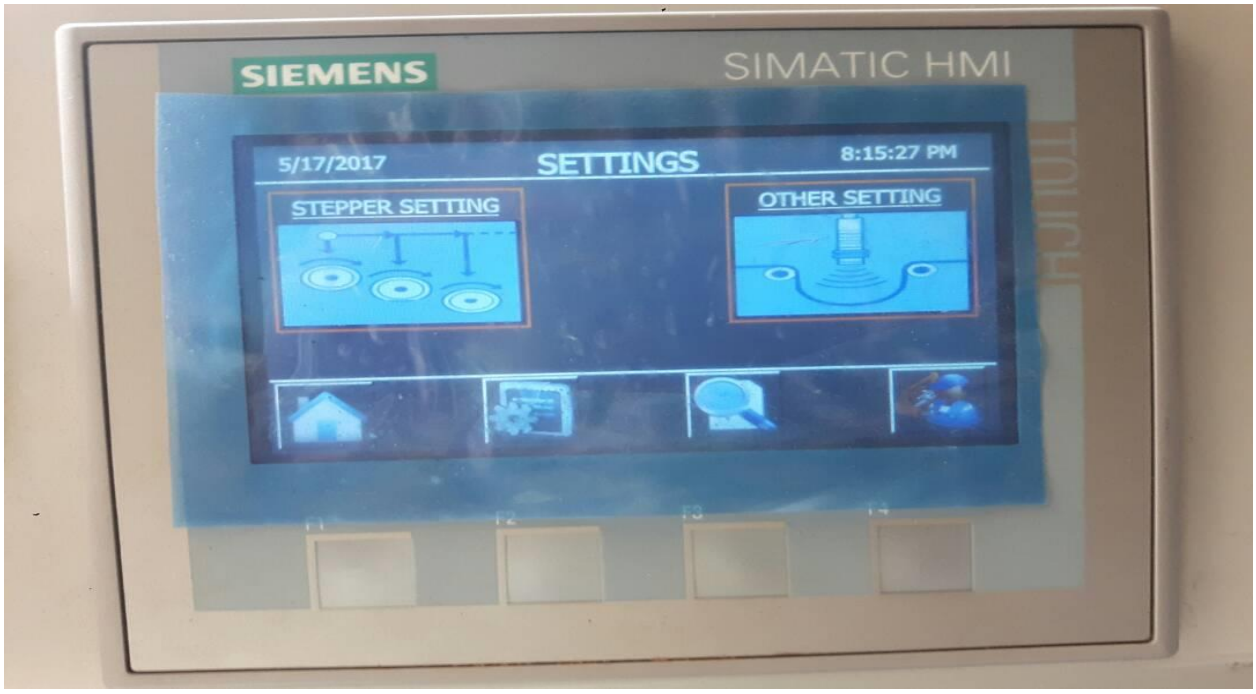

Microstep Driver
2DM556
OMRON
S8VK-C24024
POWER SUPPLY

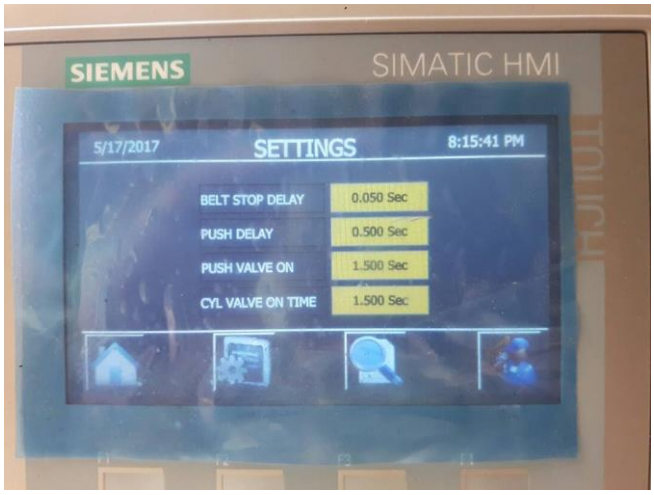

SIEMENS
SIMATIC HMI
5/17/2017
HOME
8:16:31 PM
MANUAL MODE SELECTED
AUTO MODE


SIEMENS
SIMATIC HMI
5/17/2017
IO LIST
8:16:04 PM
CPU - DIGITAL OUTPUT
BELT MOTOR PULSE
BELT MOTOR DIRECTON
CYLINDER MOTOR PULSE
CYLINDER MOTOR DIRECTON
STEPPER MOTOR 3 PULSE
STEPPER MOTOR 3 DIRECTON
PUSH VALVE
CYLINDER VALVE
<< DI


SIEMENS
SIMATIC HMI
5/17/2017
SETTINGS
8:15:32 PM
STEPPER-1 BELT
STEPPER-2 CYLINDER
STEPPER-3
SPEED
100.0 P/S
1000.0 P/S
1000.0 P/S
FRW DELAY
0.500 Sec
1.000 Sec
REV DELAY
0.500 Sec
1.000 Sec
ON TIME
6.000 Sec
1.000 Sec
BELT DIR - FRW
BELT DIR - REV


SIEMENS
SIMATIC HMI
5/17/2017
MANUAL MODE
8:16:20 PM
MANUAL MODE
VALVE
STEPPER-1 BELT
STEPPER-2 CYLINDER
STEPPER-3
PUSH
FORWARD
FORWARD
FORWARD
STRIP
REVERS
REVERS


SIEMENS
SIMATIC HMI
5/17/2017
SETTINGS
8:15:27 PM
STEPPER SETTING
OTHER SETTING


SIEMENS
SIMATIC HMI
5/17/2017
SETTINGS
8:15:41 PM
BELT STOP DELAY
0.050 Sec
PUSH DELAY
0.500 Sec
PUSH VALVE ON
1.500 Sec
CYL VALVE ON TIME
1.500 Sec


PSM86HS2A80-2P
PSM86HS2A118-2P
PSM86HS2A156-2P
MAXIMA
STEPPER MOTOR
8 Leads Motor
Series Connection
Parallel Connection
www.pbh.in

Chapter 6

Conclusion

5.1 Summary and conclusion

Nowadays, most of the control system operation in industries used PLC as a controller to control the process. It contains in the process control, transportation, domestic appliances, production lines and so on. PLC has an internal function such as timer and counter making it become sophisticated but simple in used. It also provides flexibility of control that based on the programming and can execute simple logic instruction which being used in ladder diagram.

The PLC program (ladder logic diagram) for implementing the machine has done its task in reduction of labor used for packing the food packets altogether in a box with the help of relays and the proximity sensors and using the stepper motors. It has the better processing power, memory and can handle more I/O ports, the ports are optocoupled, more robust, closed architecture and had many industrial safety features.

In summary, the aim of this project, which are to minimize the waiting time in packaging of products and to increase the production rated as well as the manufacturing till final packaging including the objectives of project have been accomplish successfully.

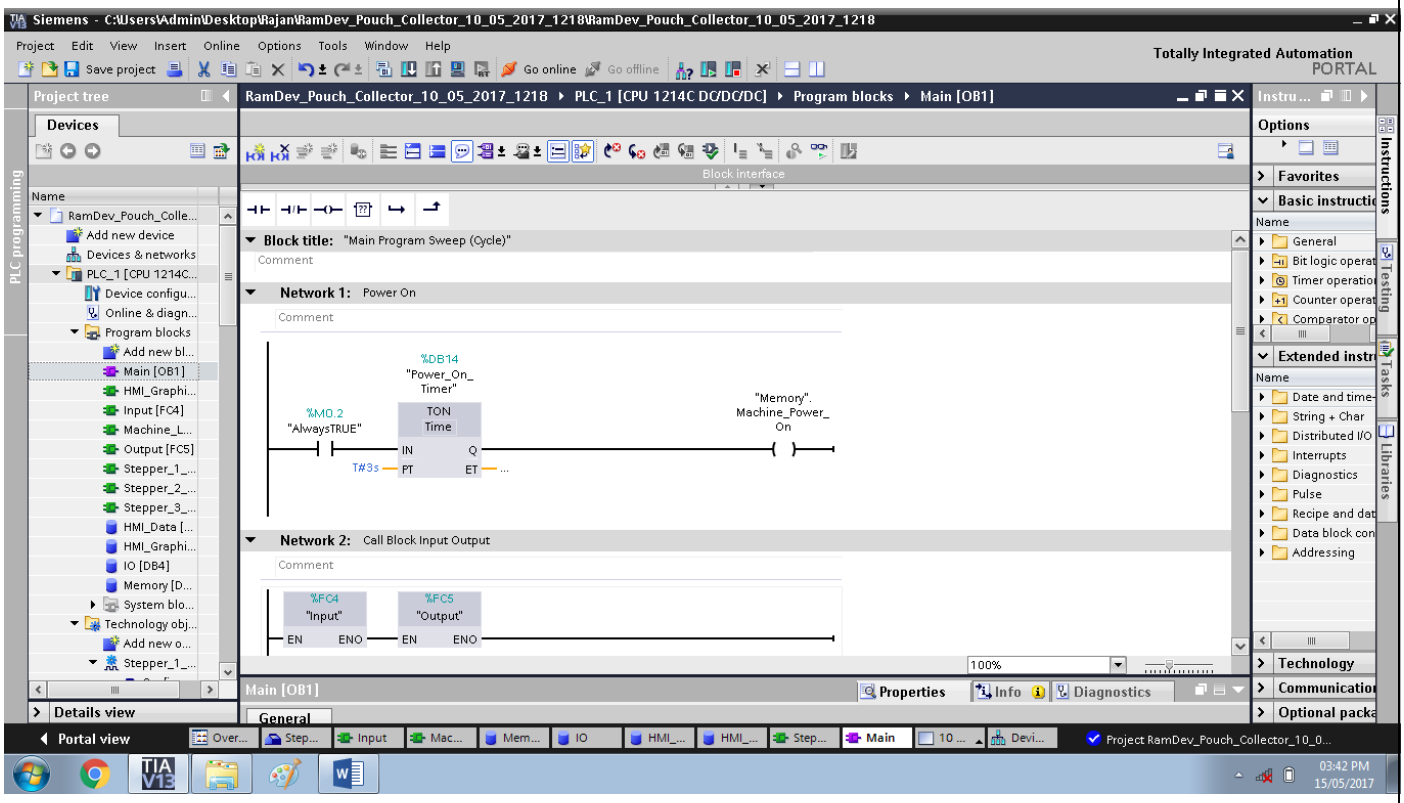
BIBLIOGRAPHY

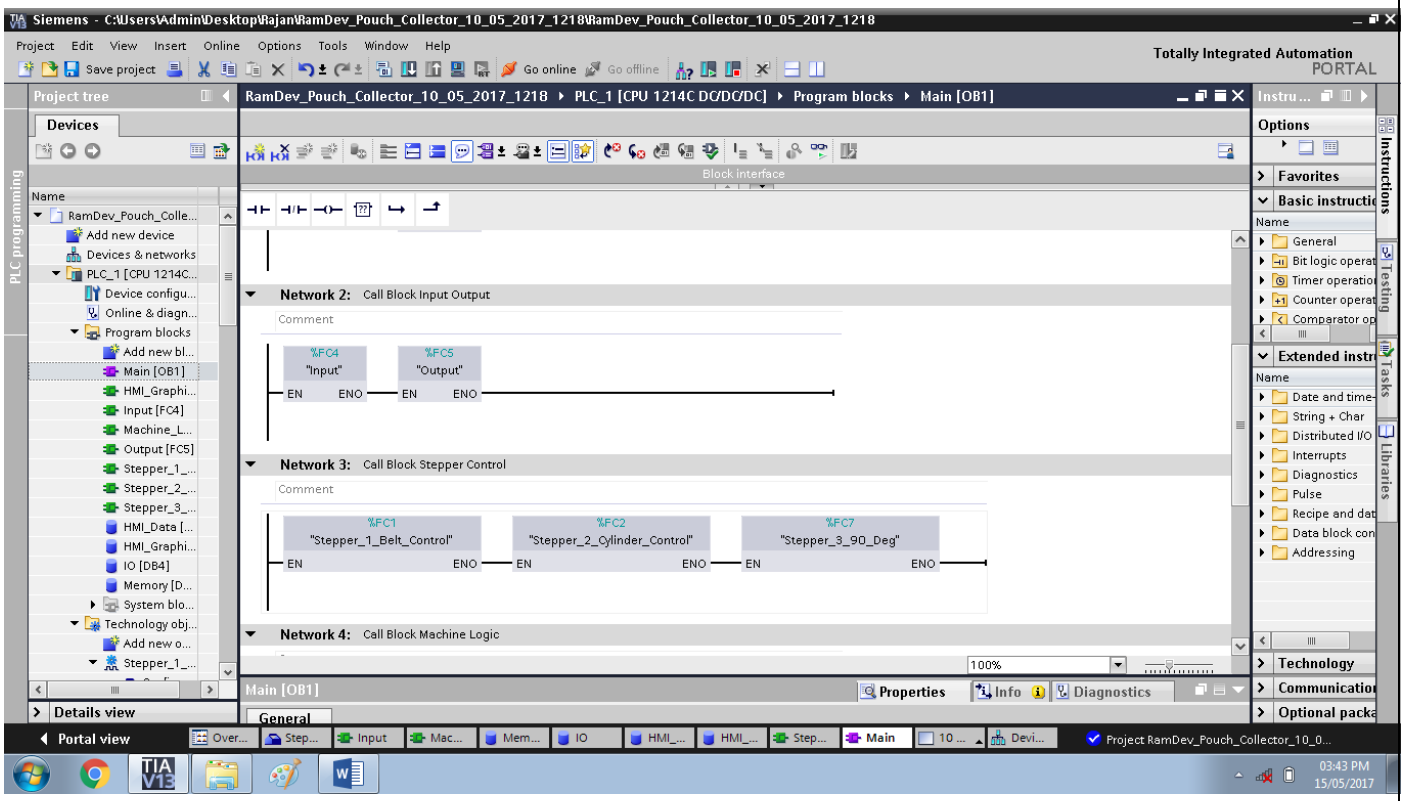
1. https://siemens.org/SimaticController/
2. https://siemens.org/S7-200/
3. https://packmechgroup.com/applications.htm

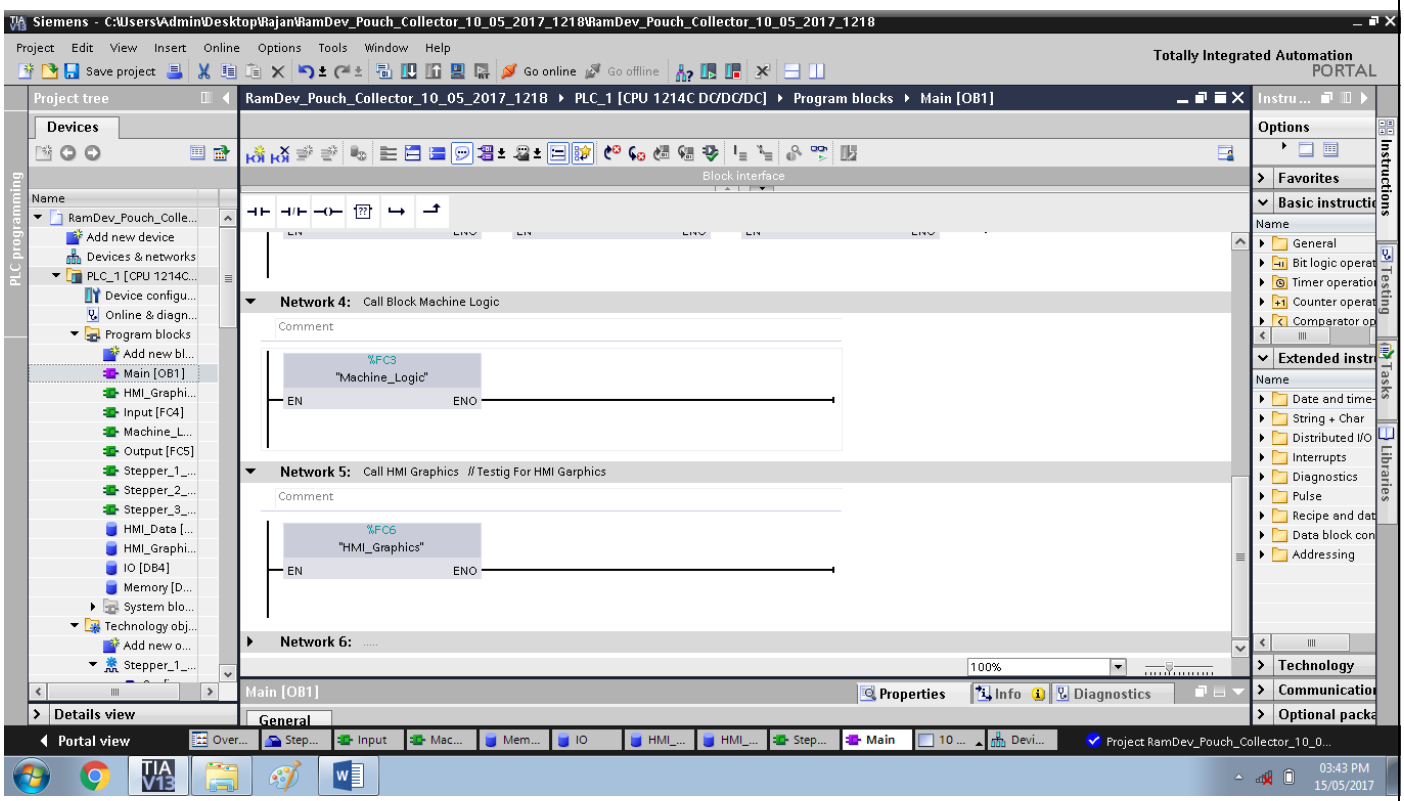
APPENDIX

CODE OF SIMEMS PLC-1214C:

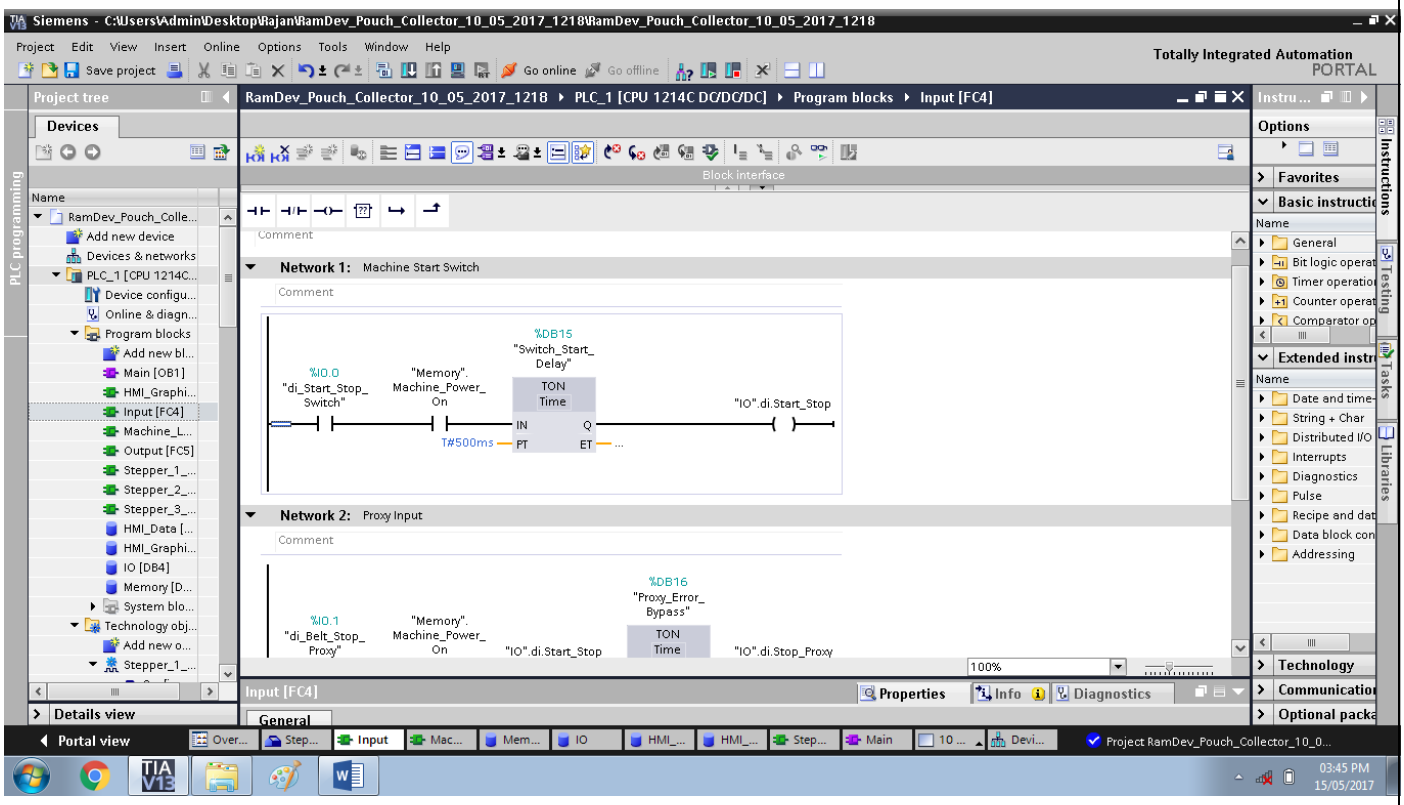
1). Main Block

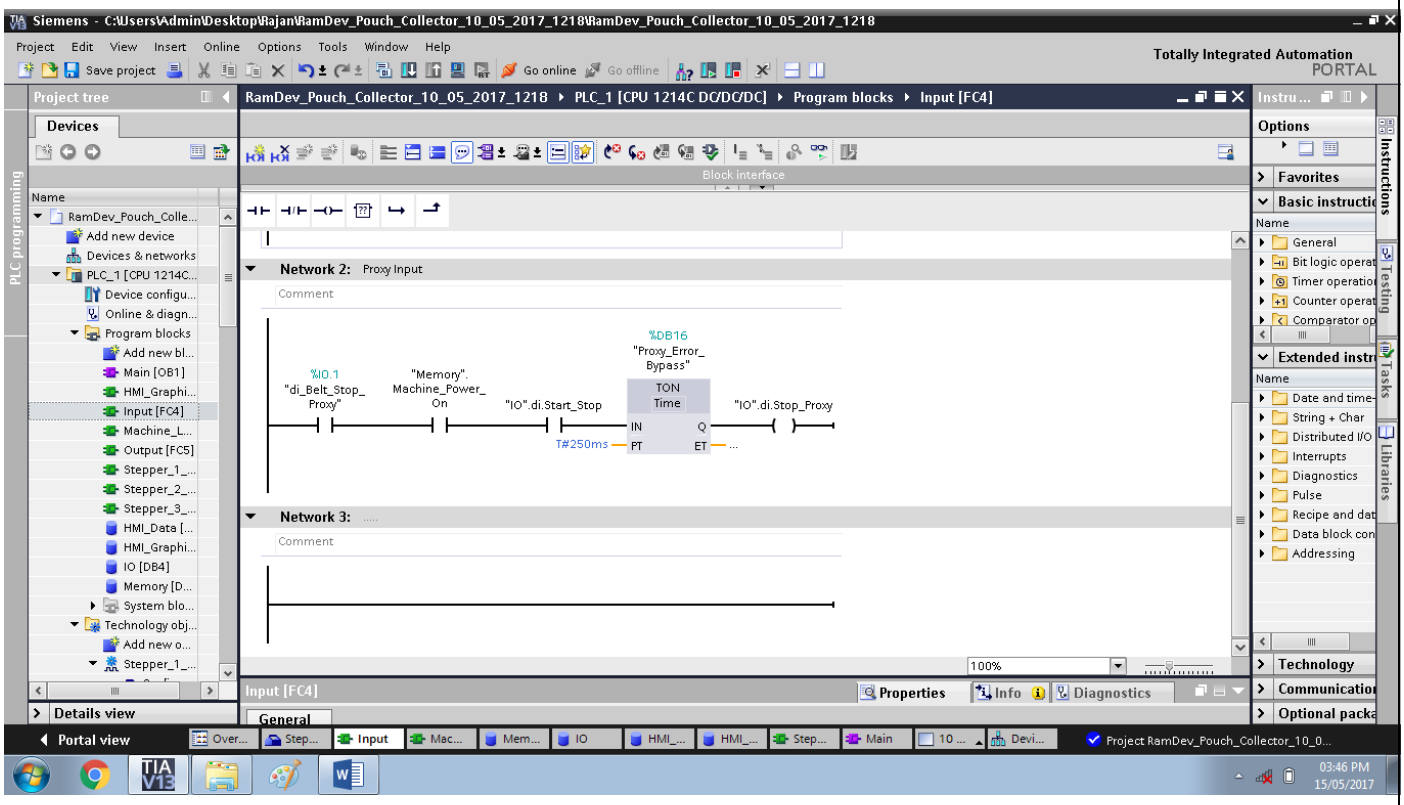



Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Main [OB1]
Block interface
Network 2: Call Block Input Output
Comment
%FC4
"Input"
%FC5
"Output"
EN ENO EN ENO
Network 3: Call Block Stepper Control
Comment
%FC1
"Stepper_1_Belt_Control"
%FC2
"Stepper_2_Cylinder_Control"
%FC7
"Stepper_3_90_Deg"
Network 4: Call Block Machine Logic
100%
Main [OB1]
Properties
Info
Diagnostics
General
Portal view
Details view
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
Instructions
Testing
Tasks
Libraries
PLC programming
03:43 PM
15/05/2017

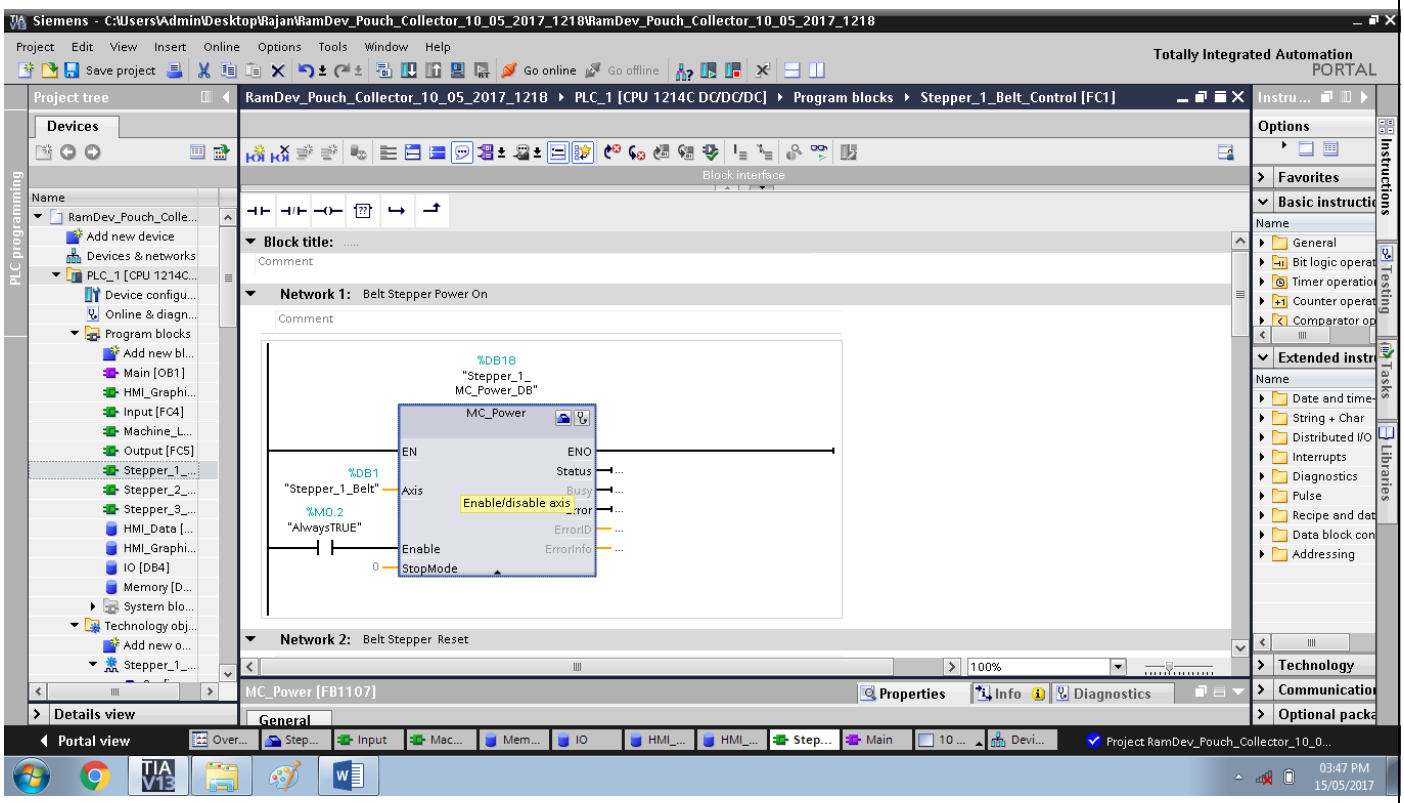

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Main [OB1]
Block interface
Network 4: Call Block Machine Logic
Comment
%FC3
"Machine_Logic"
EN ENO
Network 5: Call HMI Graphics // Testig For HMI Garphics
Comment
%FC6
"HMI_Graphics"
EN ENO
Network 6:
100%
Main [OB1]
Properties
Info
Diagnostics
General
Portal view
Details view
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
Instructions
Testing
Tasks
Libraries
PLC programming
03:43 PM
15/05/2017

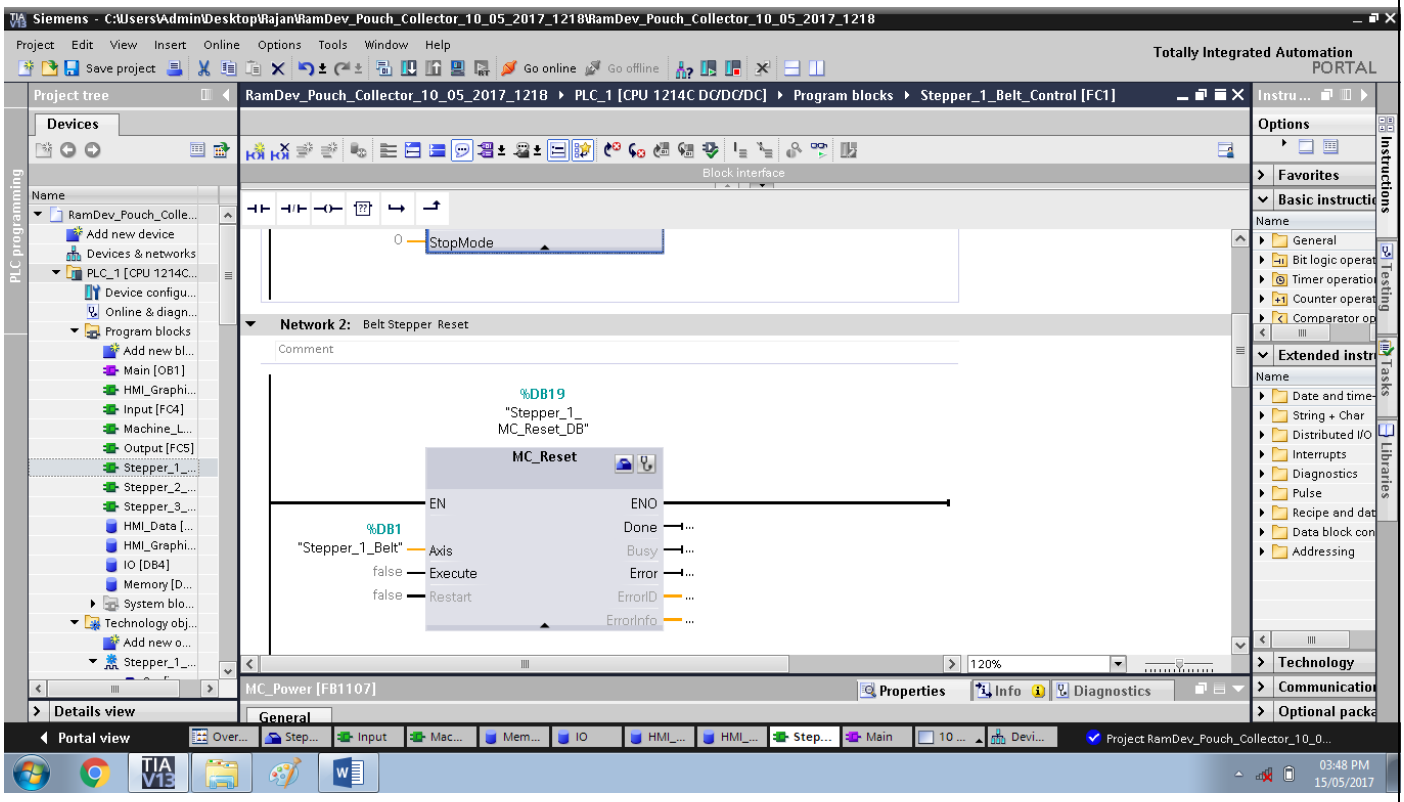
2). Input Block

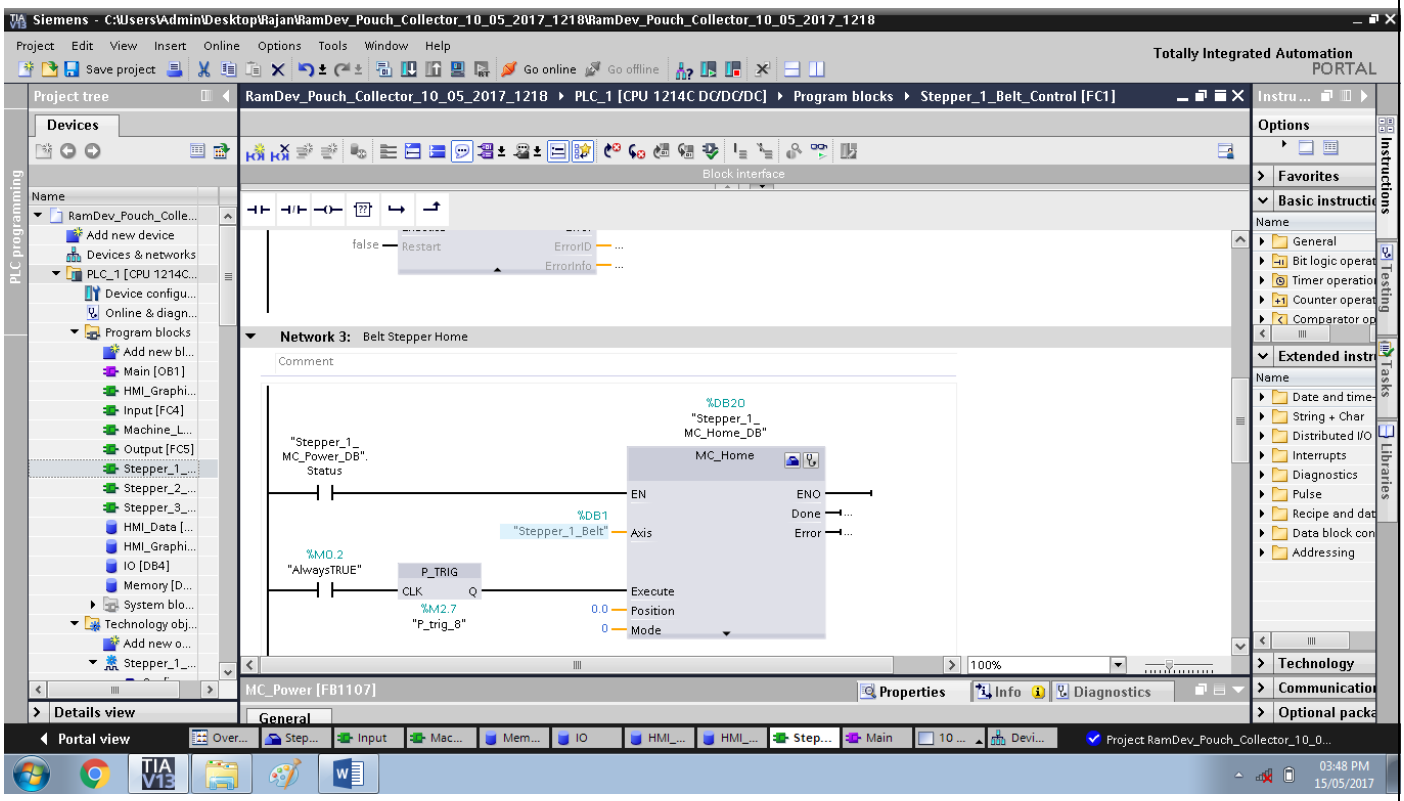


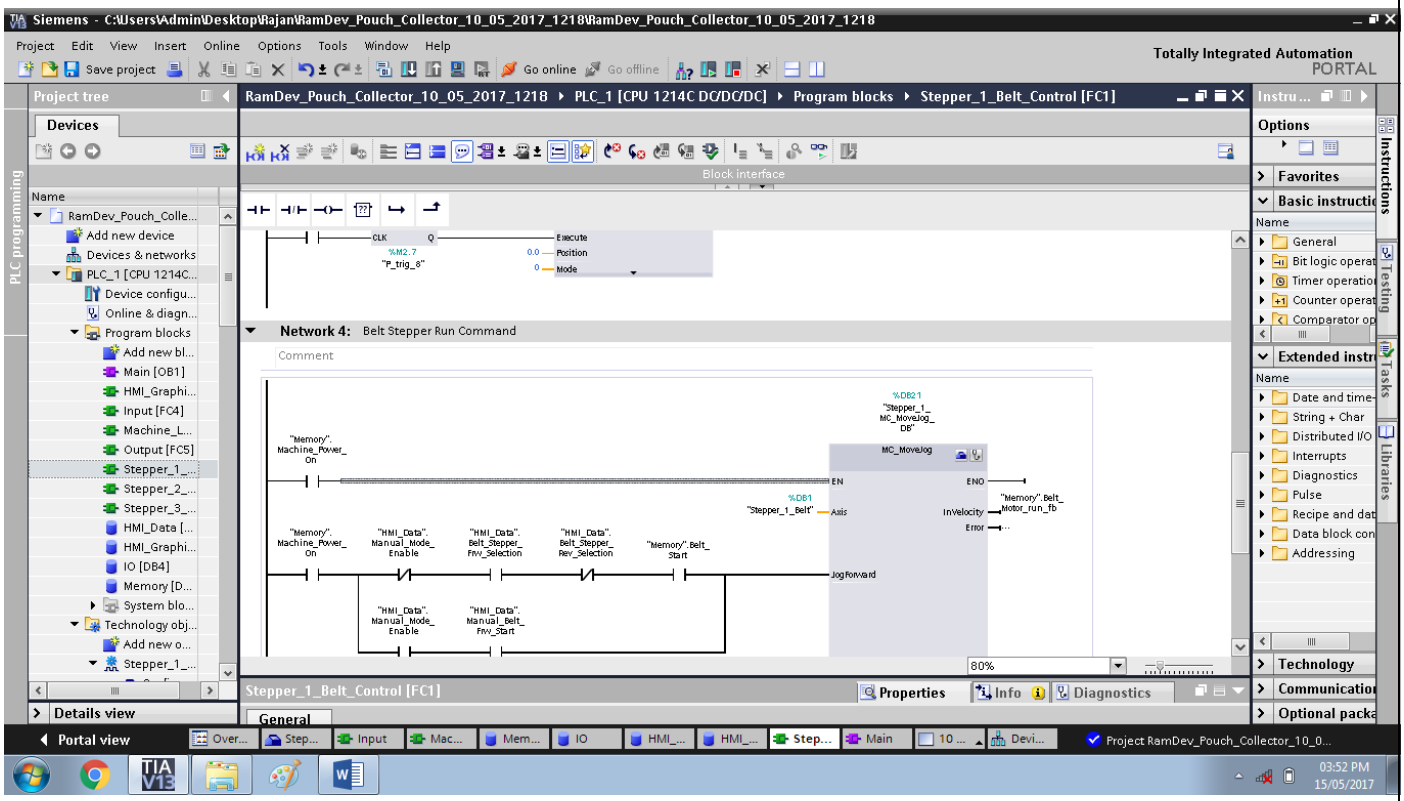


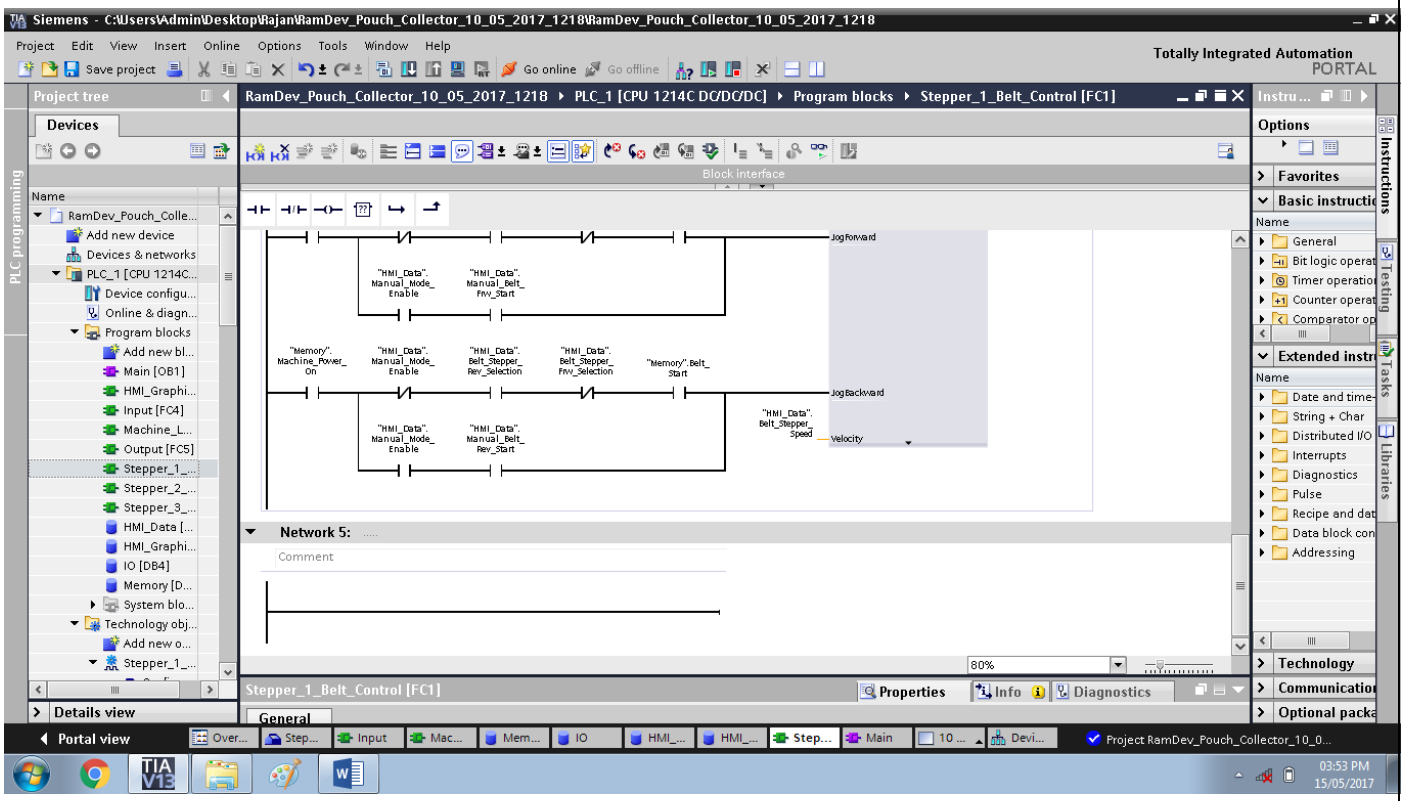
3). Stepper 1 Belt control



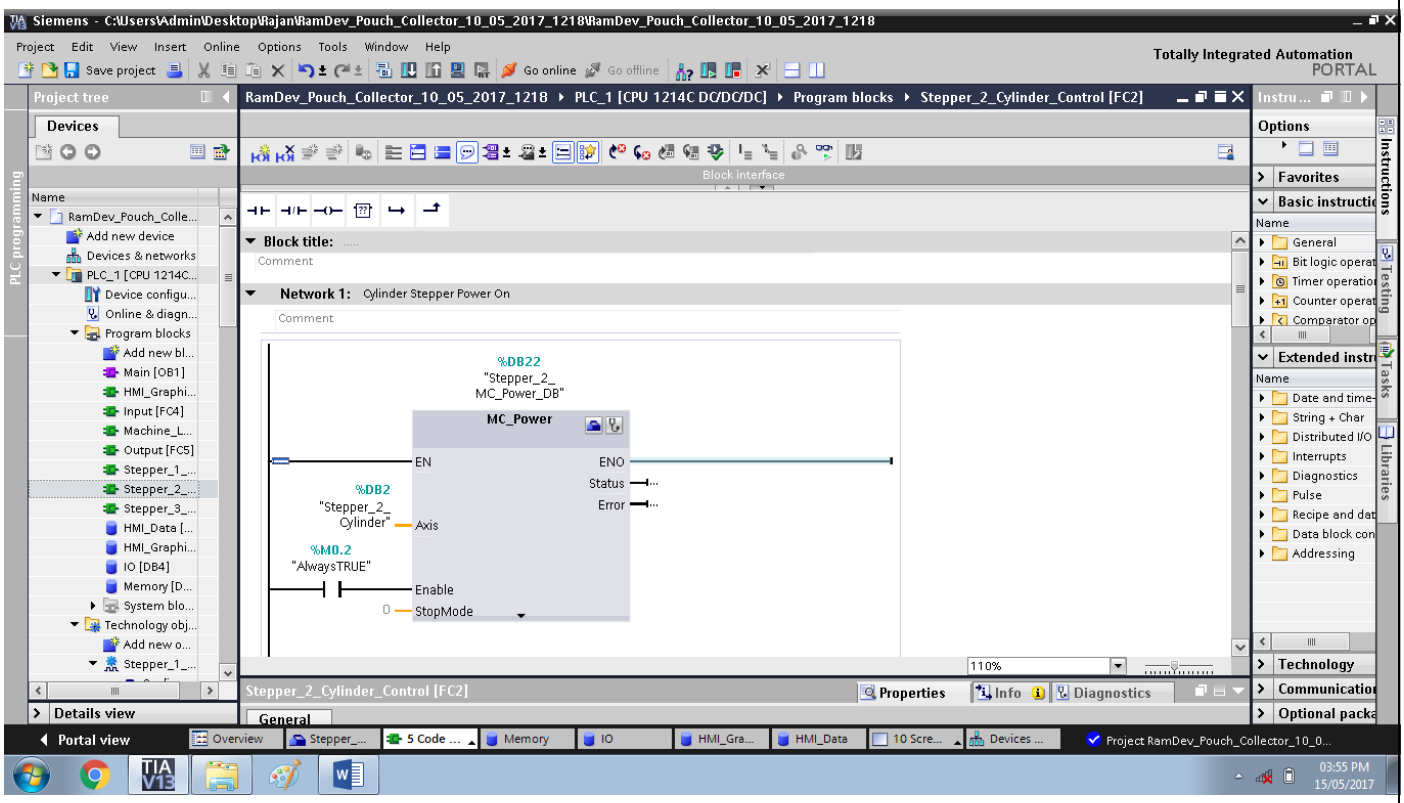


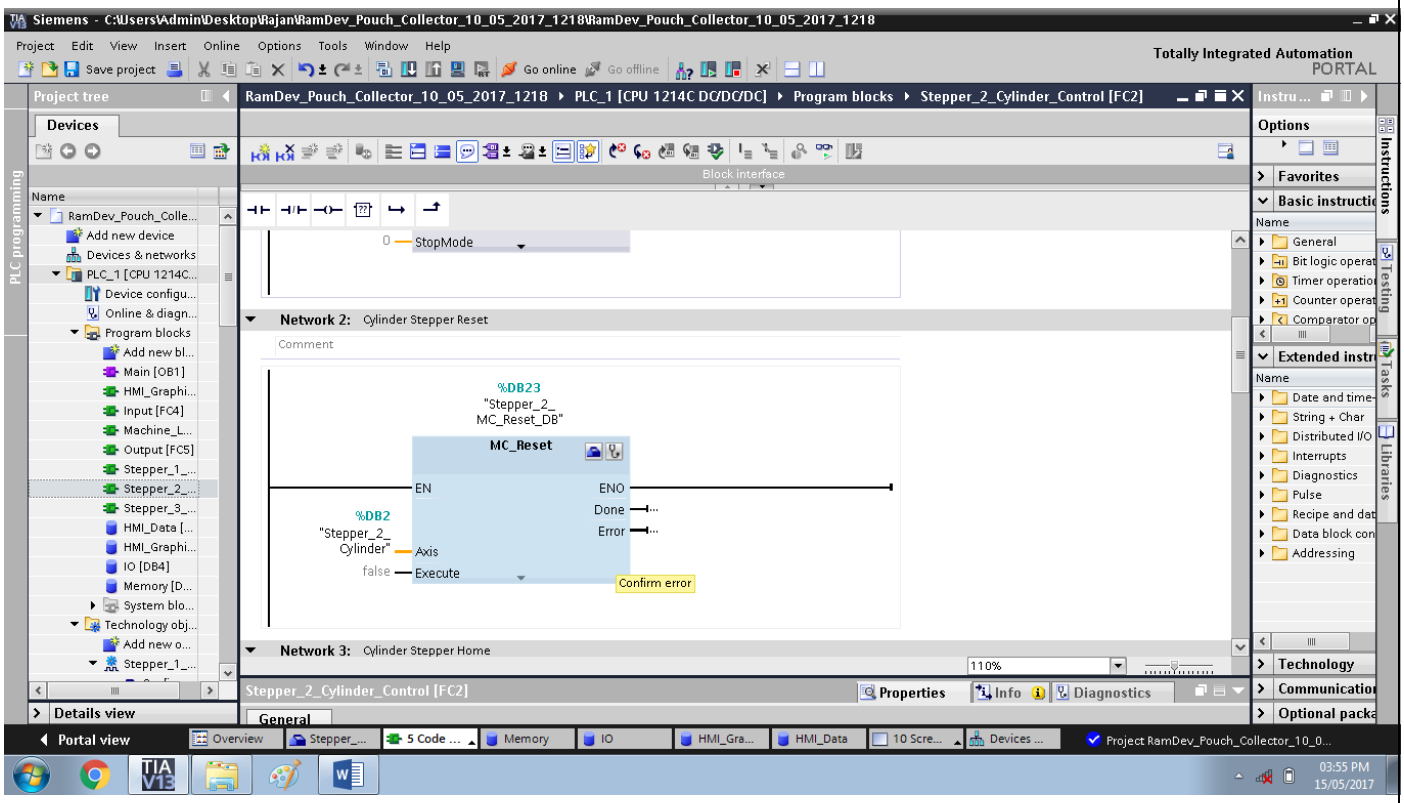

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_1_Belt_Control [FC1]
Block interface
Network 3: Belt Stepper Home
Comment
%DB20
"Stepper_1_ MC_Home_DB"
MC_Home
"Stepper_1_ MC_Power_DB". Status
EN
ENO
%DB1
"Stepper_1_Belt"
Axis
Done
Error
%M0.2
"AlwaysTRUE"
P_TRIG
CLK
Q
%M2.7
"P_trig_8"
Execute
0.0
Position
0
Mode
MC_Power [FB1107]
Properties
Info
Diagnostics
General
Portal view
Details view
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
03:48 PM
15/05/2017

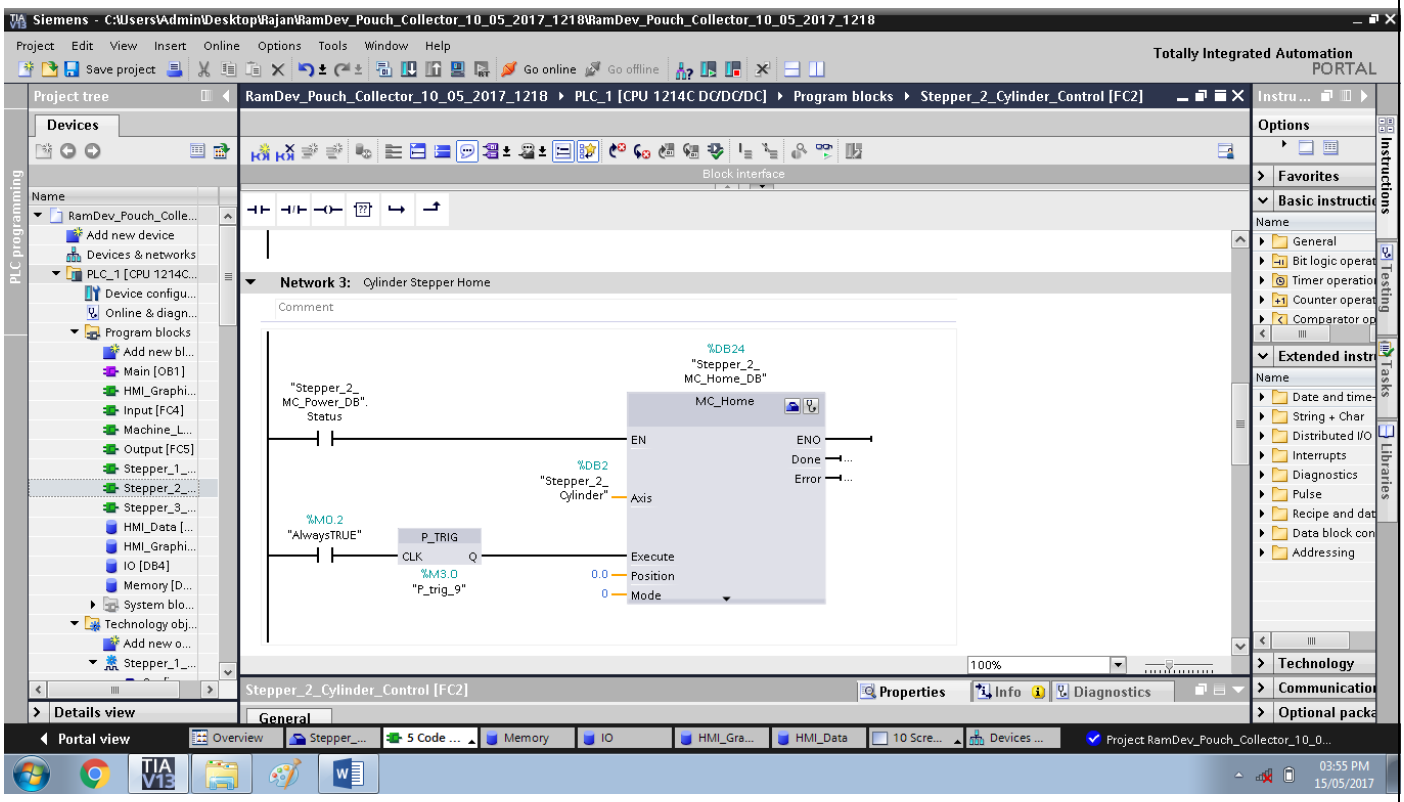

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_1_Belt_Control [FC1]
Block interface
CLK
Q
%M2.7
"P_trig_8"
Execute
0.0
Position
0
Mode
Network 4: Belt Stepper Run Command
Comment
%DB21
"Stepper_1_ MC_MoveJog_ DB"
MC_MoveJog
"Memory". Machine_Power_ On
EN
ENO
%DB1
"Stepper_1_Belt"
Axis
InVelocity
Error
"Memory".Belt_ Motor_run_fb
"Memory". Machine_Power_ On
"HMI_Data". Manual_Mode_ Enable
"HMI_Data". Belt_Stepper_ Fwv_Selection
"HMI_Data". Belt_Stepper_ Rev_Selection
"Memory".Belt_ Start
JogForward
"HMI_Data". Manual_Mode_ Enable
"HMI_Data". Manual_Belt_ Fwv_Start
Stepper_1_Belt_Control [FC1]
Properties
Info
Diagnostics
General
Portal view
Details view
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
03:52 PM
15/05/2017

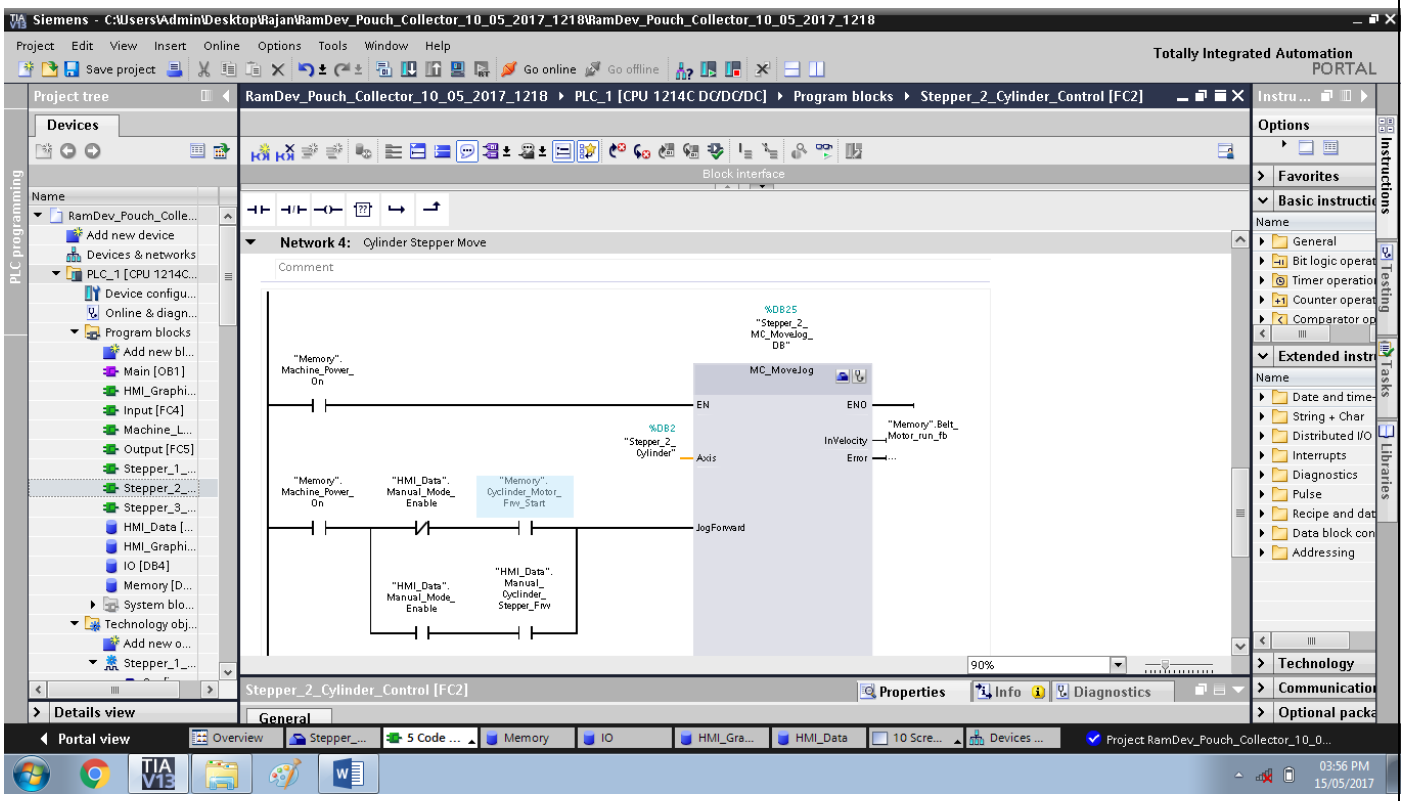


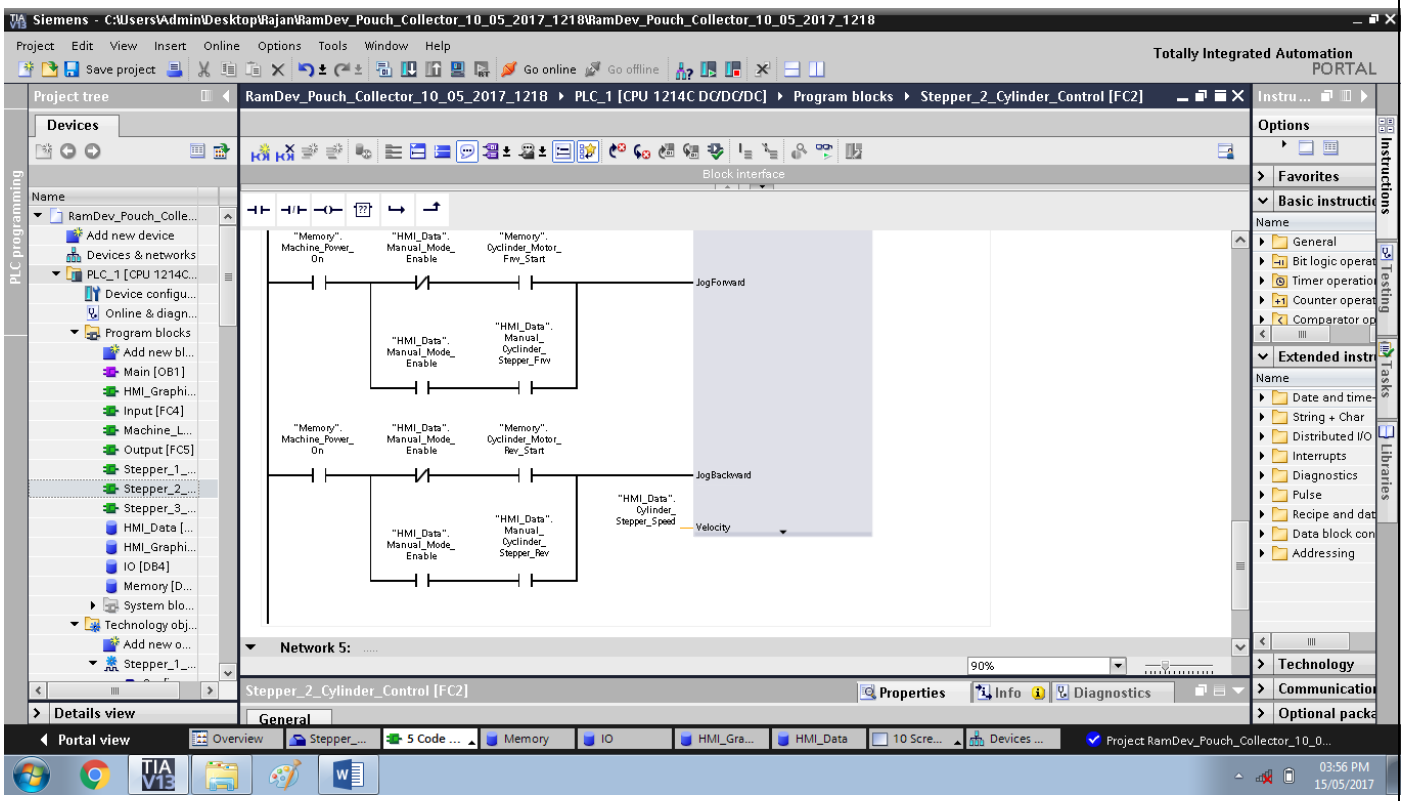
4). Stepper 2 Cylinder control



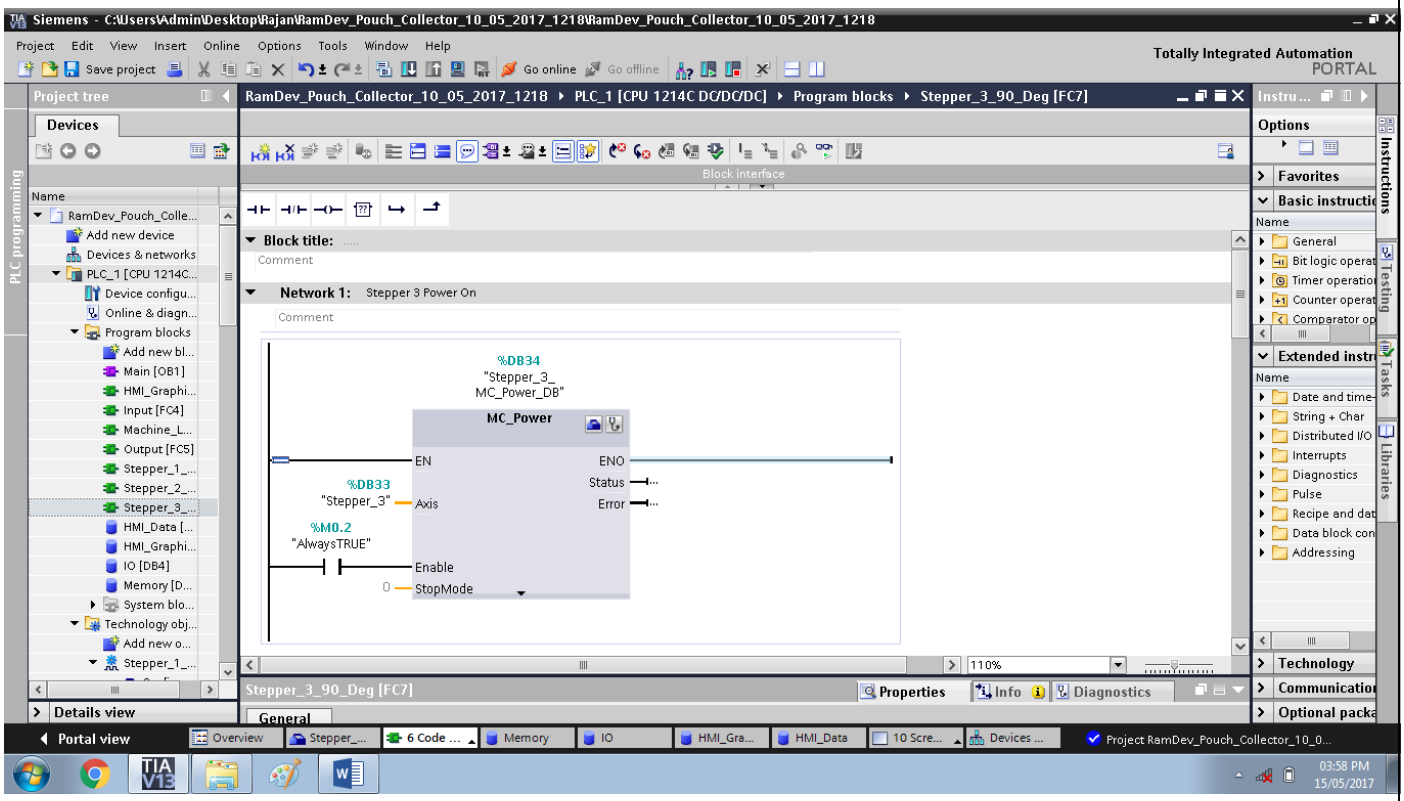

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_2_Cylinder_Control [FC2]
Block interface
0 StopMode
Network 2: Cylinder Stepper Reset
Comment
%DB23
"Stepper_2_ MC_Reset_DB"
MC_Reset
EN
ENO
%DB2
"Stepper_2_ Cylinder"
Axis
Done
Error
false
Execute
Confirm error
Network 3: Cylinder Stepper Home
Stepper_2_Cylinder_Control [FC2]
Properties
Info
Diagnostics
General
Portal view
Overview
Details view

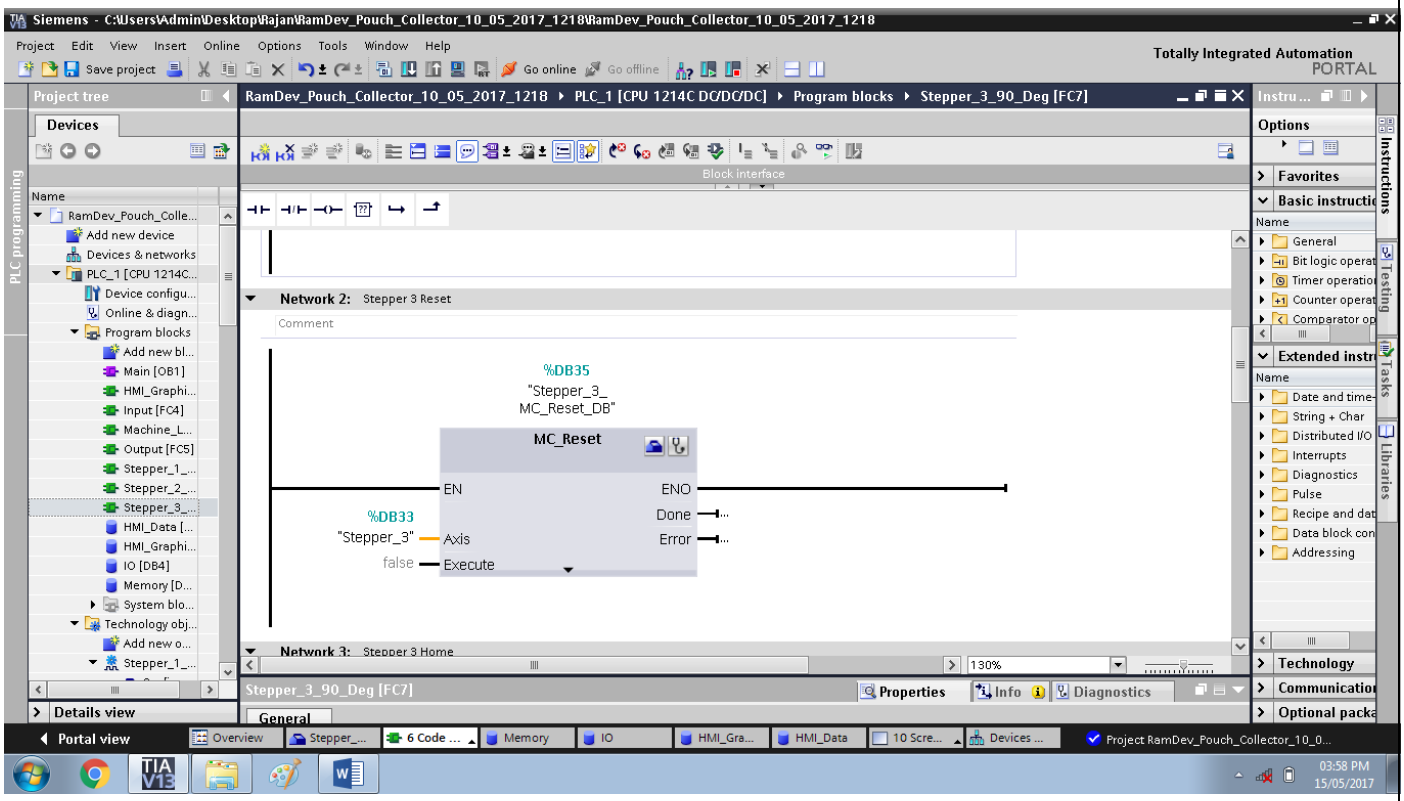

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_2_Cylinder_Control [FC2]
Block interface
Network 3: Cylinder Stepper Home
Comment
%DB24
"Stepper_2_ MC_Home_DB"
"Stepper_2_ MC_Power_DB". Status
MC_Home
EN
ENO
Done
Error
%DB2
"Stepper_2_ Cylinder"
Axis
%M0.2
"AlwaysTRUE"
P_TRIG
CLK
Q
%M3.0
"P_trig_9"
Execute
0.0
Position
0
Mode
Stepper_2_Cylinder_Control [FC2]
Properties
Info
Diagnostics
General
Portal view
Overview
Details view

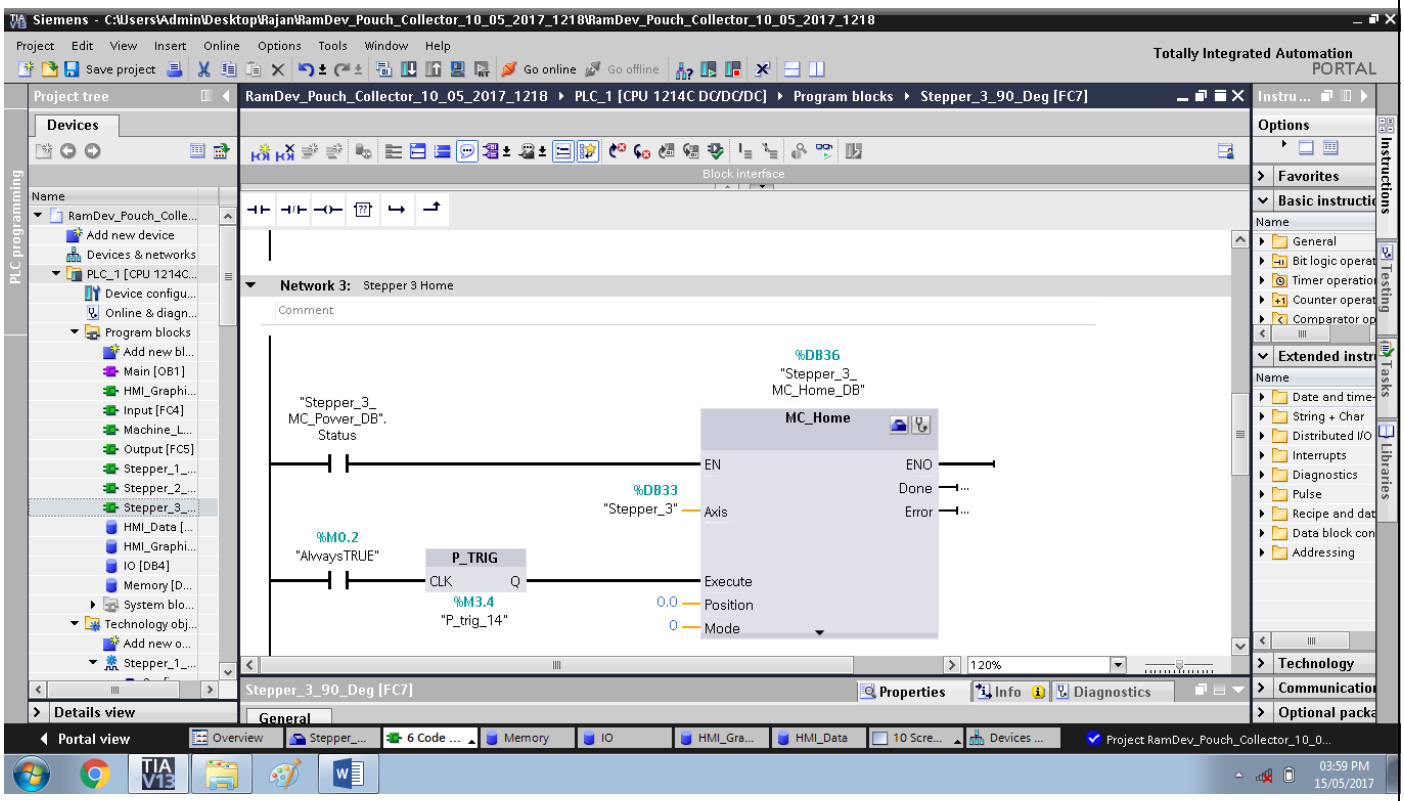

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_2_Cylinder_Control [FC2]
Network 4: Cylinder Stepper Move
Comment
%DB25 "Stepper_2_MC_MoveJog_DB"
MC_MoveJog
"Memory".Machine_Power_On
EN
ENO
%DB2 "Stepper_2_Cylinder"
Axis
InVelocity
Error
"Memory".Belt_Motor_run_fb
"HMI_Data".Manual_Mode_Enable
"Memory".Cyclinder_Motor_Frw_Start
JogForward
"HMI_Data".Manual_Cyclinder_Stepper_Frw
Stepper_2_Cylinder_Control [FC2]
Properties
Info
Diagnostics
General
Portal view
Overview
Project RamDev_Pouch_Collector_10_0...
03:56 PM
15/05/2017

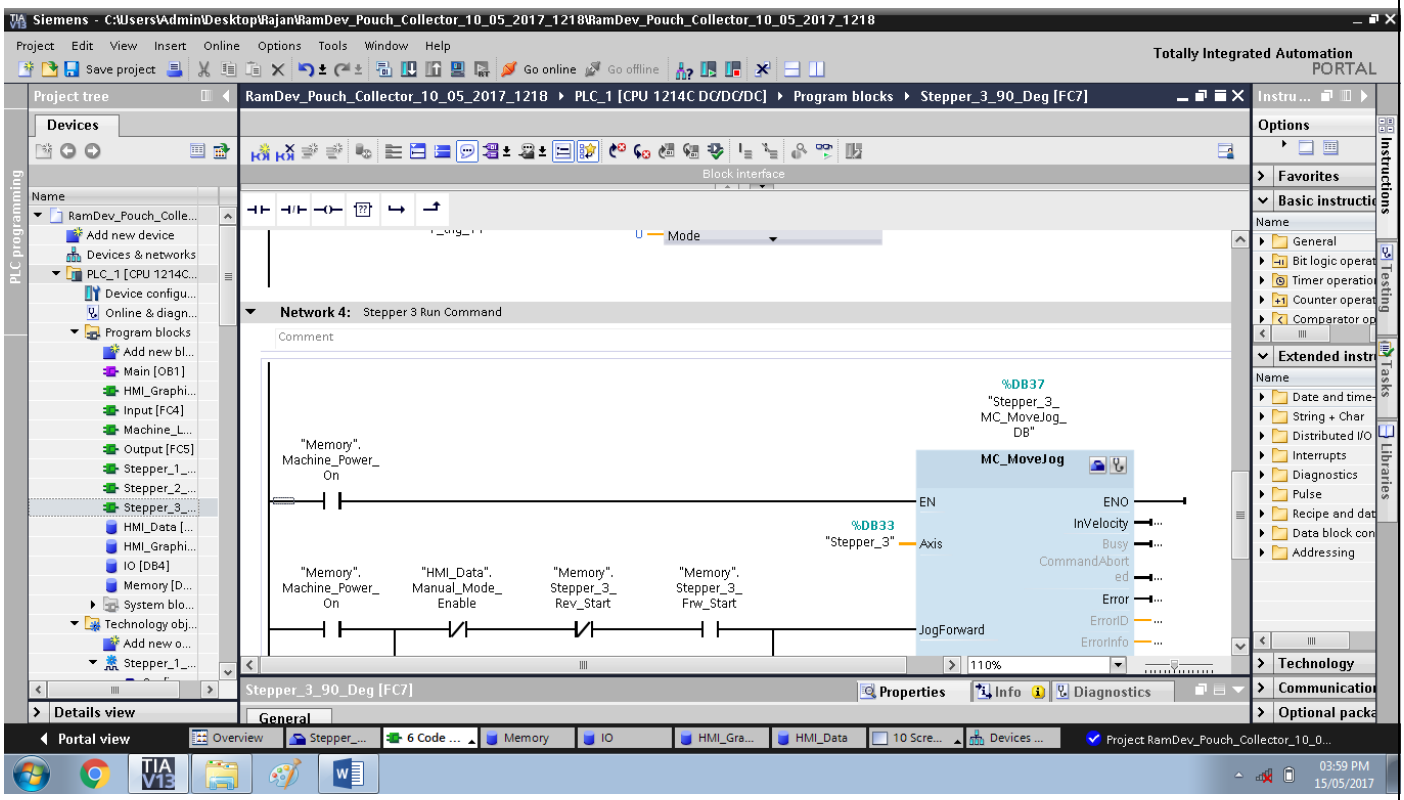

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_2_Cylinder_Control [FC2]
"Memory".Machine_Power_On
"HMI_Data".Manual_Mode_Enable
"Memory".Cyclinder_Motor_Frw_Start
JogForward
"HMI_Data".Manual_Cyclinder_Stepper_Frw
"Memory".Cyclinder_Motor_Rev_Start
JogBackward
"HMI_Data".Cylinder_Stepper_Speed
Velocity
"HMI_Data".Manual_Cyclinder_Stepper_Rev
Network 5:
Stepper_2_Cylinder_Control [FC2]
Properties
Info
Diagnostics
General
Portal view
Overview
Project RamDev_Pouch_Collector_10_0...
03:56 PM
15/05/2017

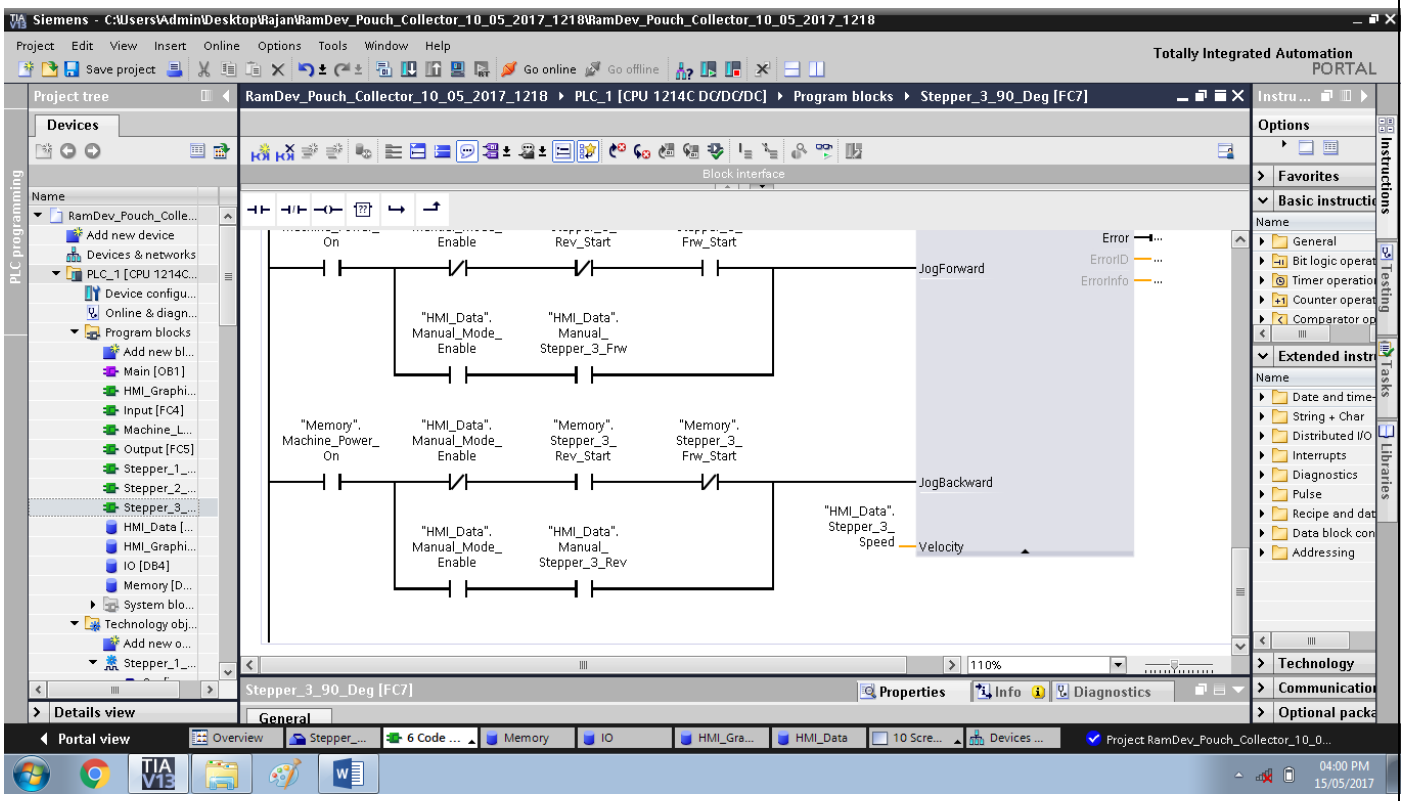
5). Stepper 3 Dual Rod Control



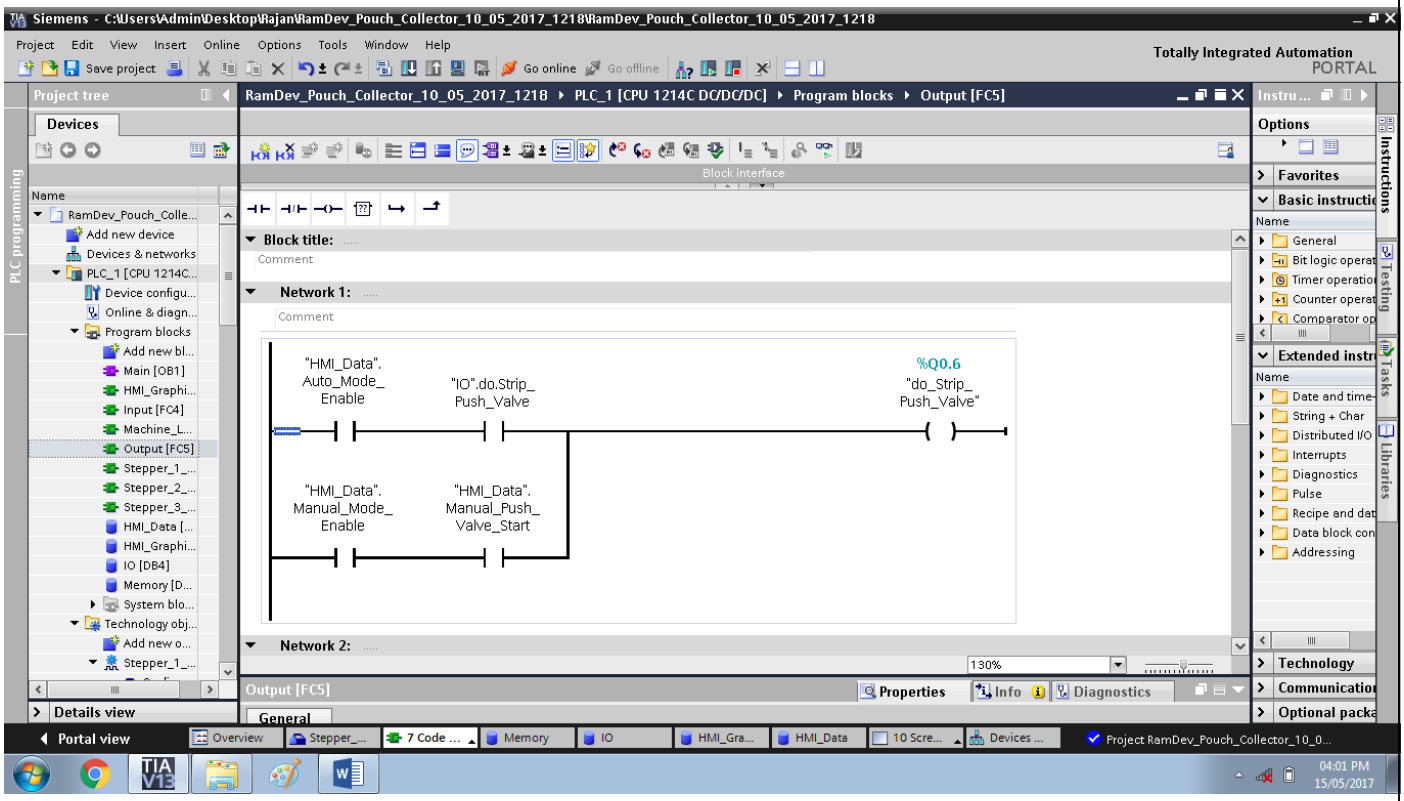


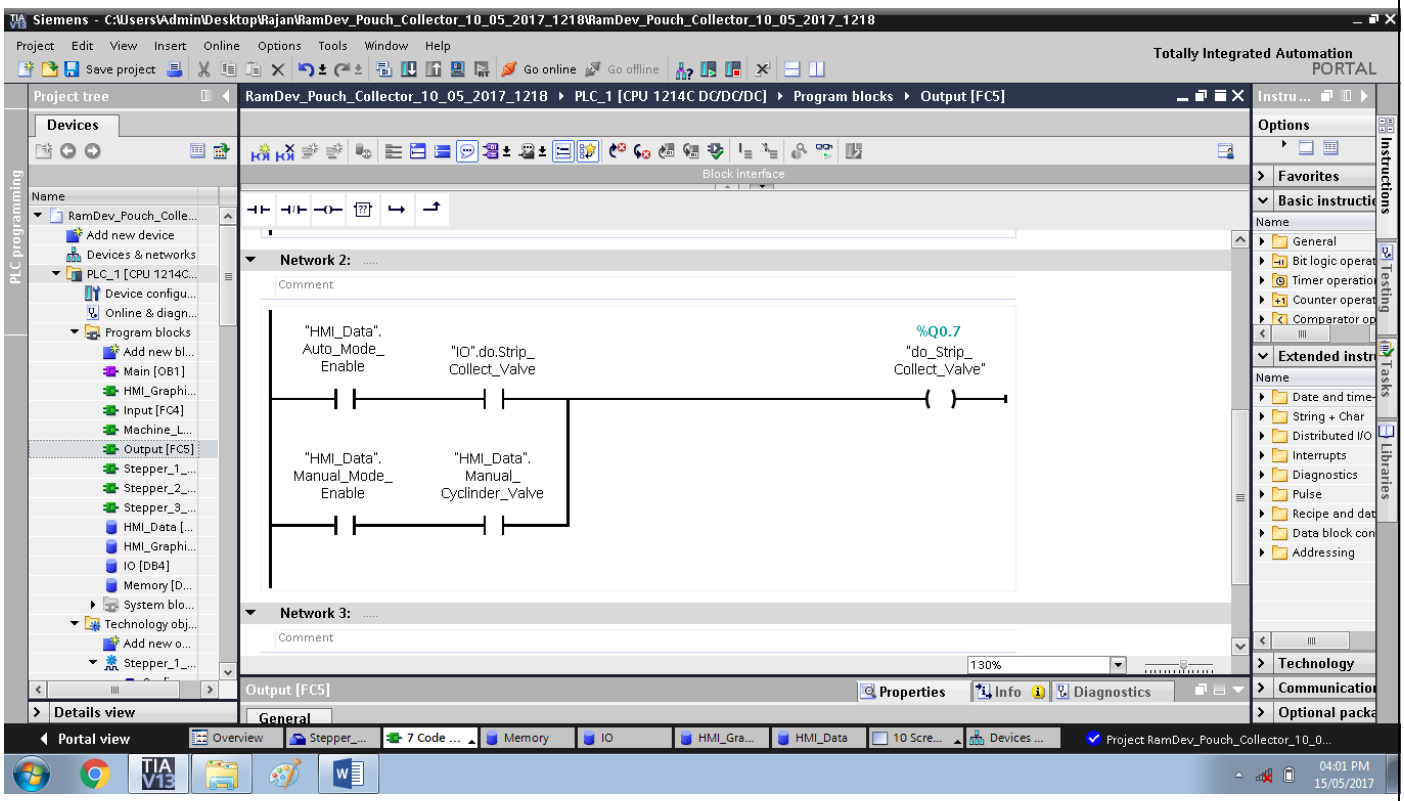

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_3_90_Deg [FC7]
Block interface
Network 3: Stepper 3 Home
Comment
%DB36
"Stepper_3_ MC_Home_DB"
"Stepper_3_ MC_Power_DB". Status
MC_Home
EN
ENO
%DB33
"Stepper_3"
Axis
Done
Error
%M0.2
"AlwaysTRUE"
P_TRIG
CLK
Q
Execute
%M3.4
"P_trig_14"
0.0
Position
0
Mode
120%
Stepper_3_90_Deg [FC7]
Properties
Info
Diagnostics
General
Portal view
Overview
Details view
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
03:59 PM
15/05/2017


Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Stepper_3_90_Deg [FC7]
Block interface
Mode
Network 4: Stepper 3 Run Command
Comment
%DB37
"Stepper_3_ MC_MoveJog_ DB"
"Memory". Machine_Power_ On
MC_MoveJog
EN
ENO
%DB33
"Stepper_3"
Axis
InVelocity
Busy
CommandAborted
Error
ErrorID
ErrorInfo
"Memory". Machine_Power_ On
"HMI_Data". Manual_Mode_ Enable
"Memory". Stepper_3_ Rev_Start
"Memory". Stepper_3_ Frw_Start
JogForward
110%
Stepper_3_90_Deg [FC7]
Properties
Info
Diagnostics
General
Portal view
Overview
Details view
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
03:59 PM
15/05/2017

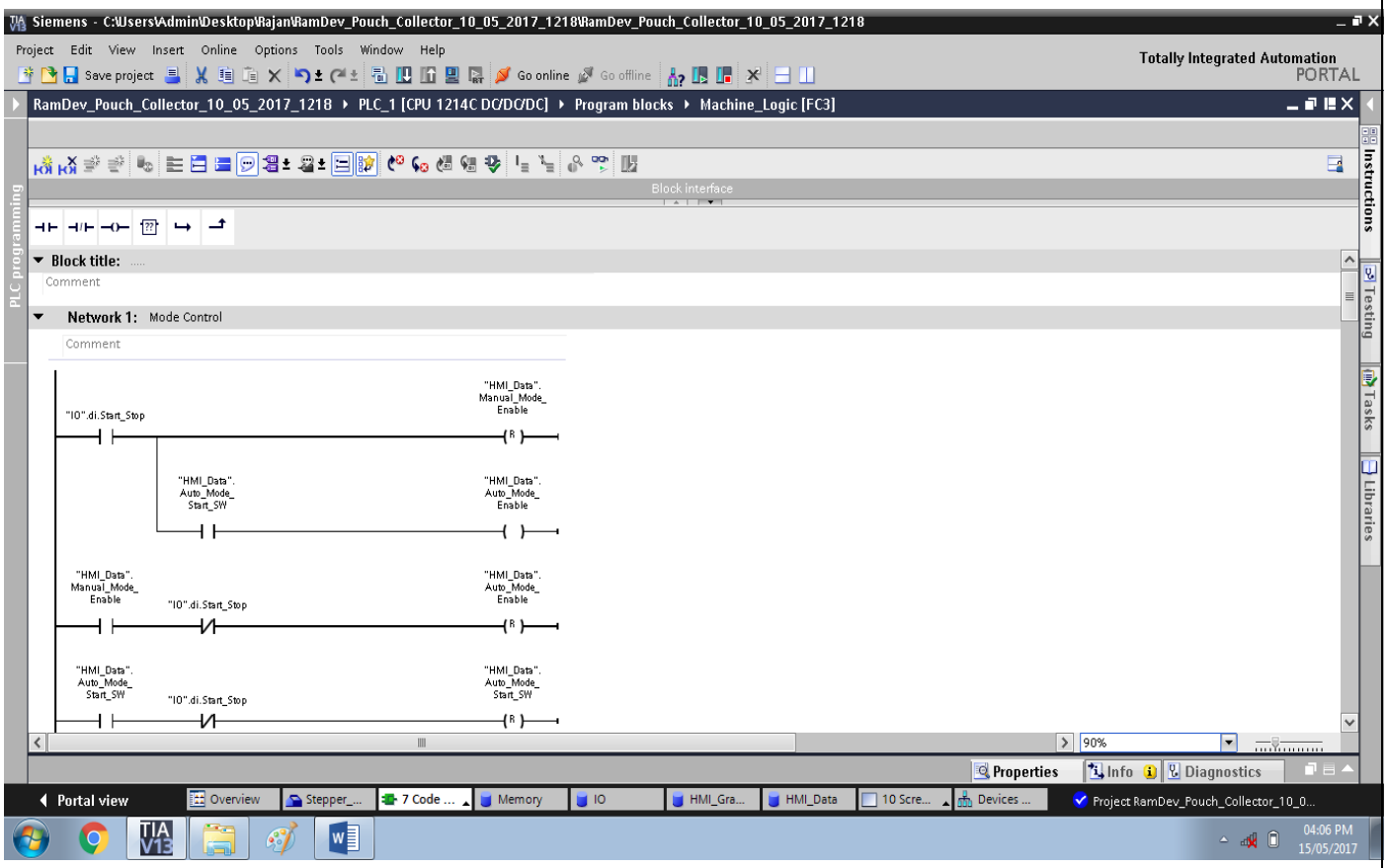


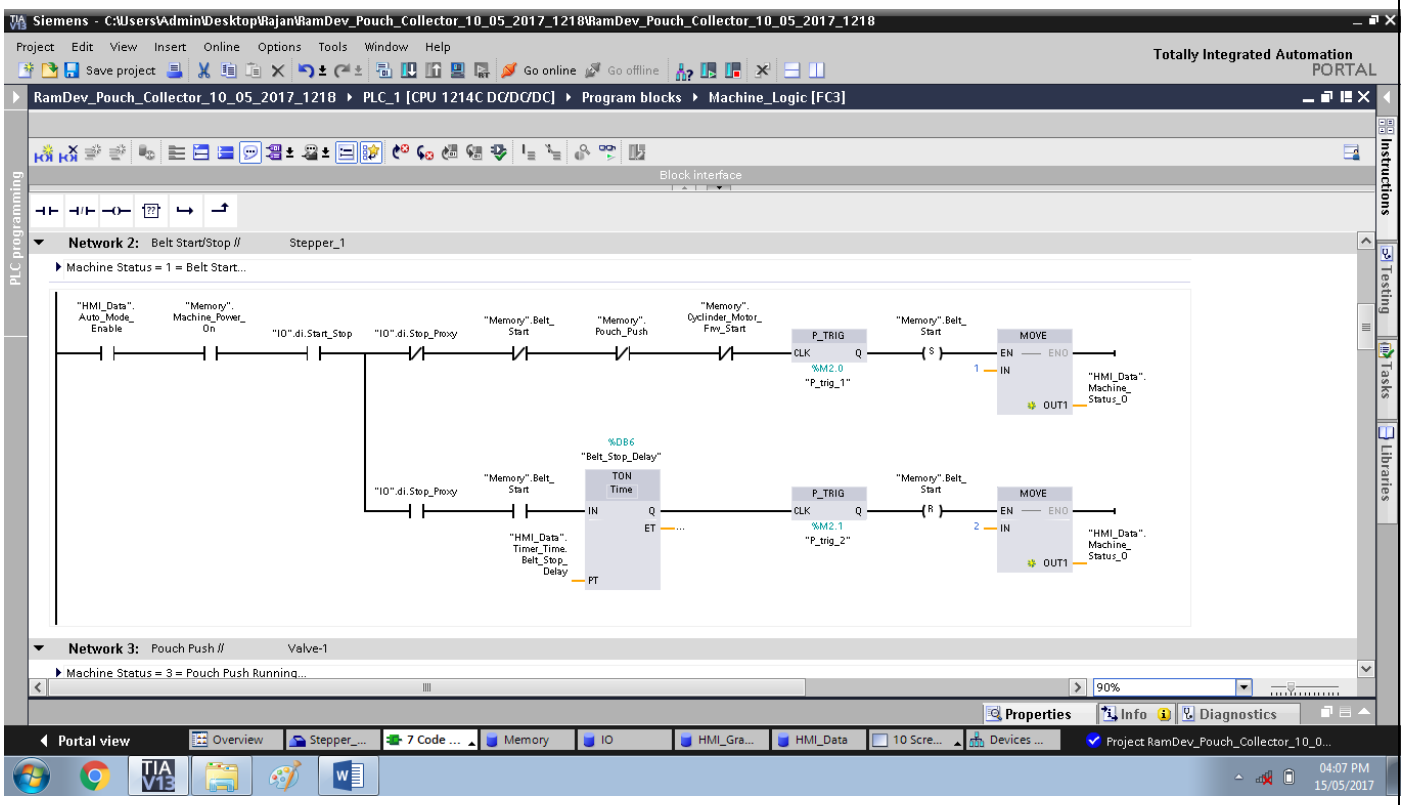
6). Output Block

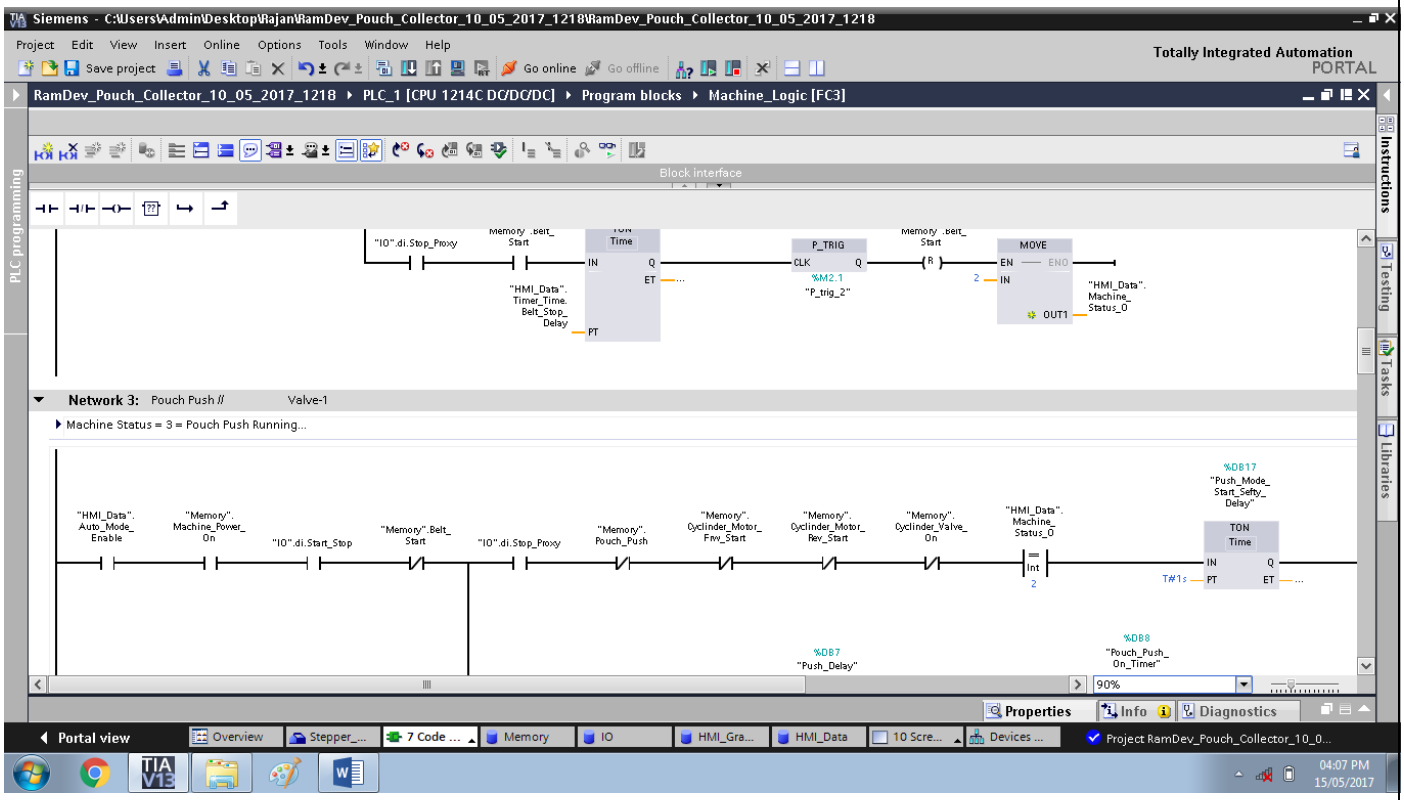


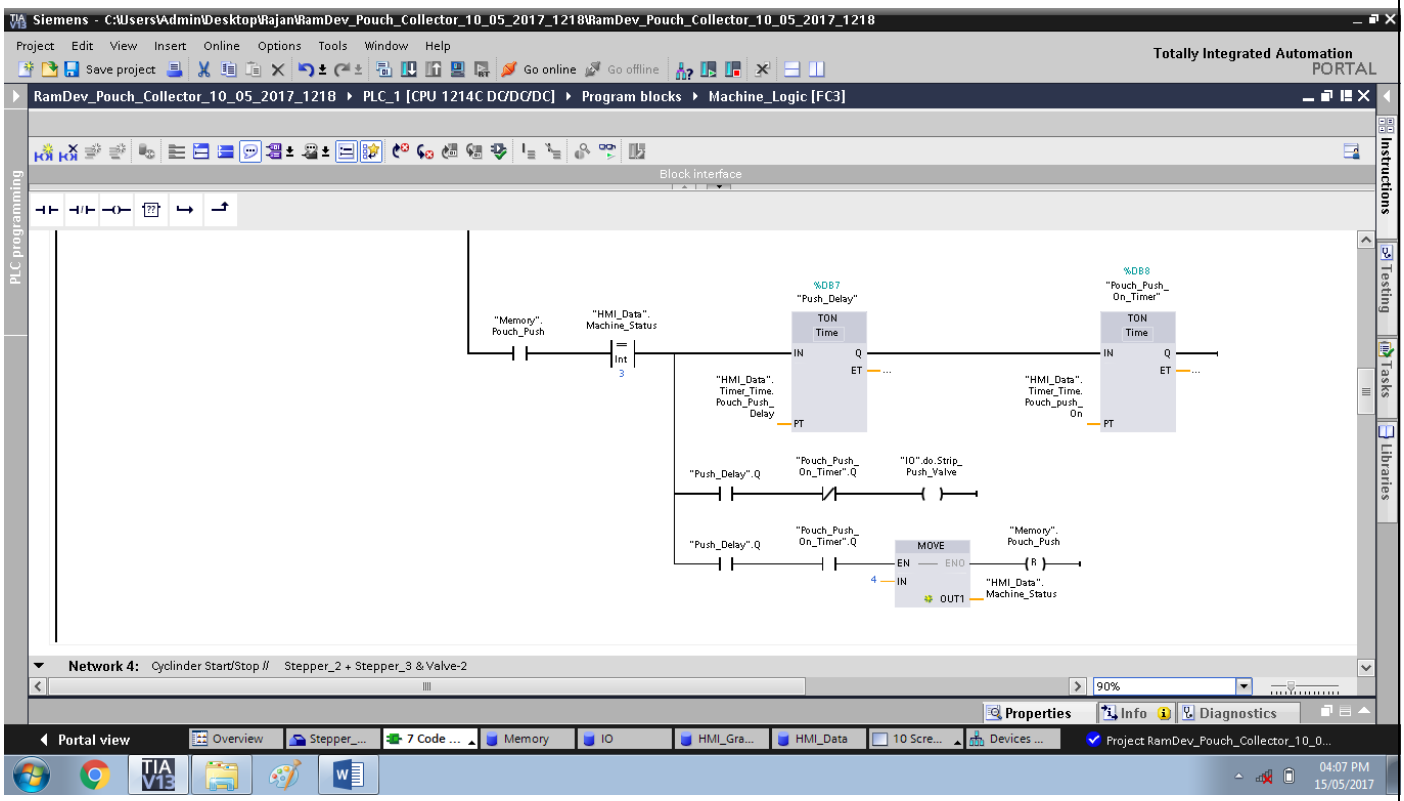

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Output [FC5]
Block interface
Network 2:
Comment
"HMI_Data".
Auto_Mode_
Enable
"IO".do.Strip_
Collect_Valve
%Q0.7
"do_Strip_
Collect_Valve"
"HMI_Data".
Manual_Mode_
Enable
"HMI_Data".
Manual_
Cyclinder_Valve
Network 3:
Comment
130%
Output [FC5]
Properties
Info
Diagnostics
General
Details view
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:01 PM
15/05/2017
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
Instructions
Testing
Tasks
Libraries
PLC programming

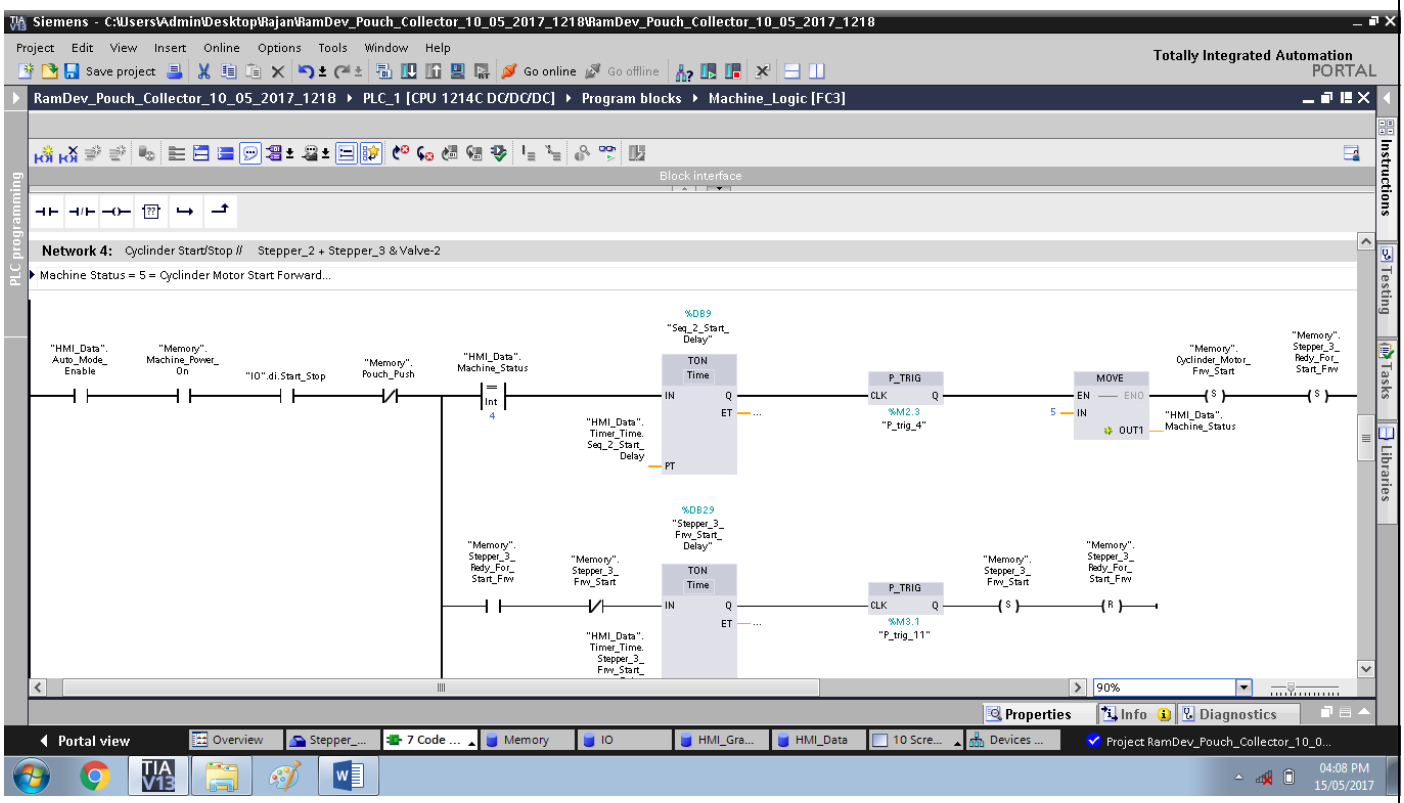
7). Main Logic

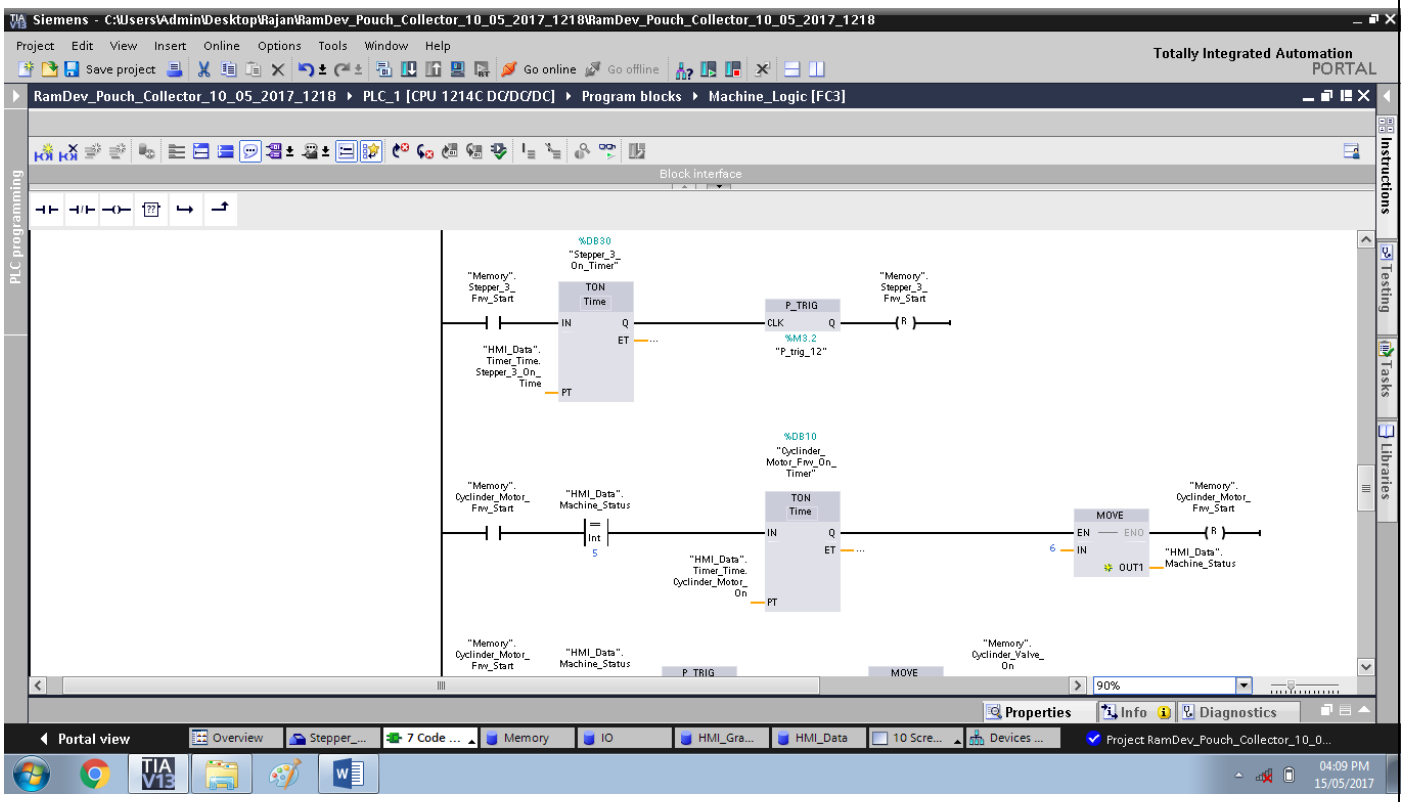

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
Block title:
Comment
Network 1: Mode Control
Comment
"IO".di.Start_Stop
"HMI_Data".Manual_Mode_Enable
"HMI_Data".Auto_Mode_Start_SW
"HMI_Data".Auto_Mode_Enable
"IO".di.Start_Stop
"HMI_Data".Auto_Mode_Start_SW
"IO".di.Start_Stop
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:06 PM
15/05/2017

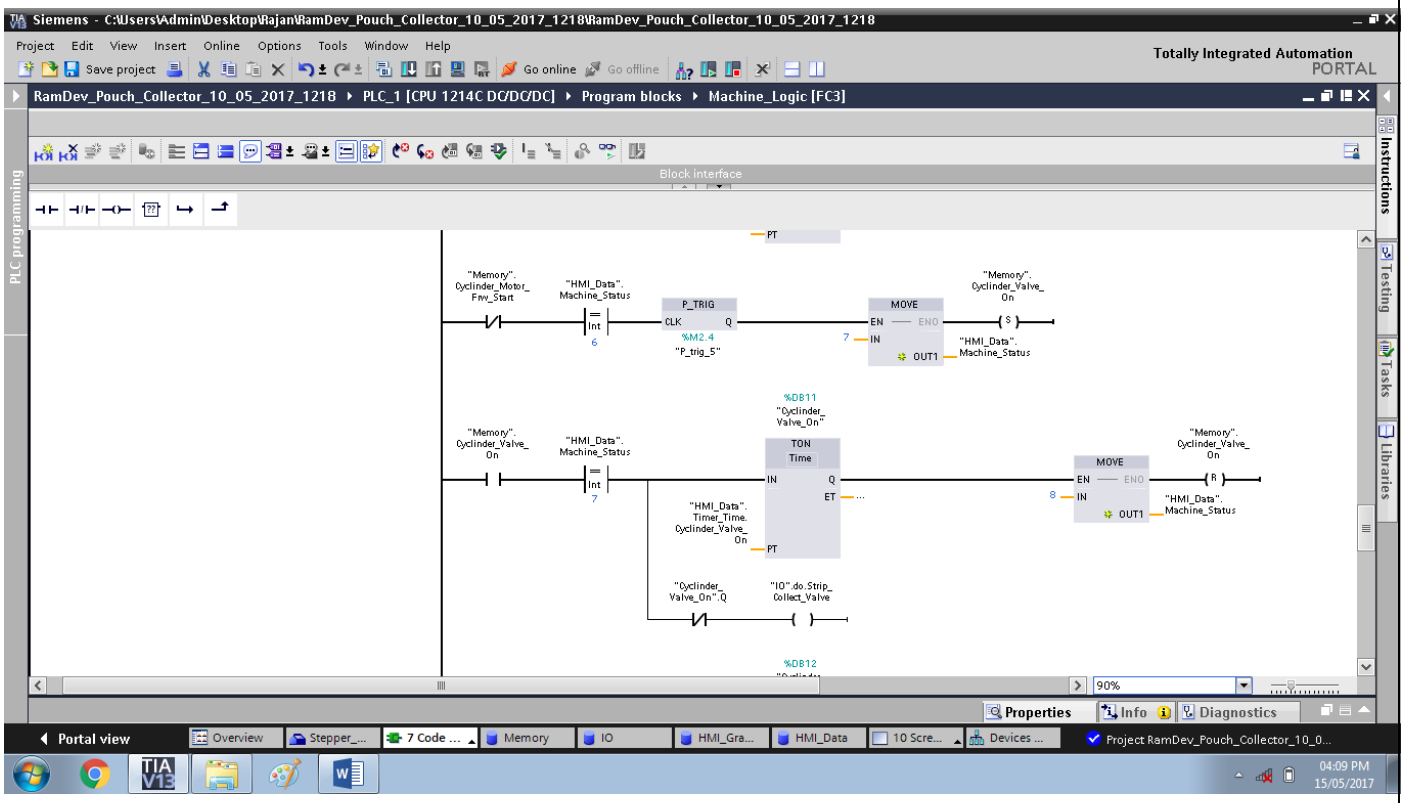

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
Network 2: Belt Start/Stop // Stepper_1
Machine Status = 1 = Belt Start...
"HMI_Data".Auto_Mode_Enable
"Memory".Machine_Power_On
"IO".di.Start_Stop
"IO".di.Stop_Proxy
"Memory".Belt_Start
"Memory".Pouch_Push
"Memory".Cyclinder_Motor_Fwv_Start
P_TRIG
CLK
Q
%M2.0
"P_trig_1"
"Memory".Belt_Start
MOVE
EN
ENO
1
IN
OUT1
"HMI_Data".Machine_Status_0
%DB6
"Belt_Stop_Delay"
TON
Time
IN
Q
ET
PT
"HMI_Data".Timer_Time.Belt_Stop_Delay
"IO".di.Stop_Proxy
"Memory".Belt_Start
P_TRIG
CLK
Q
%M2.1
"P_trig_2"
"Memory".Belt_Start
MOVE
EN
ENO
2
IN
OUT1
"HMI_Data".Machine_Status_0
Network 3: Pouch Push // Valve-1
Machine Status = 3 = Pouch Push Running...
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:07 PM
15/05/2017

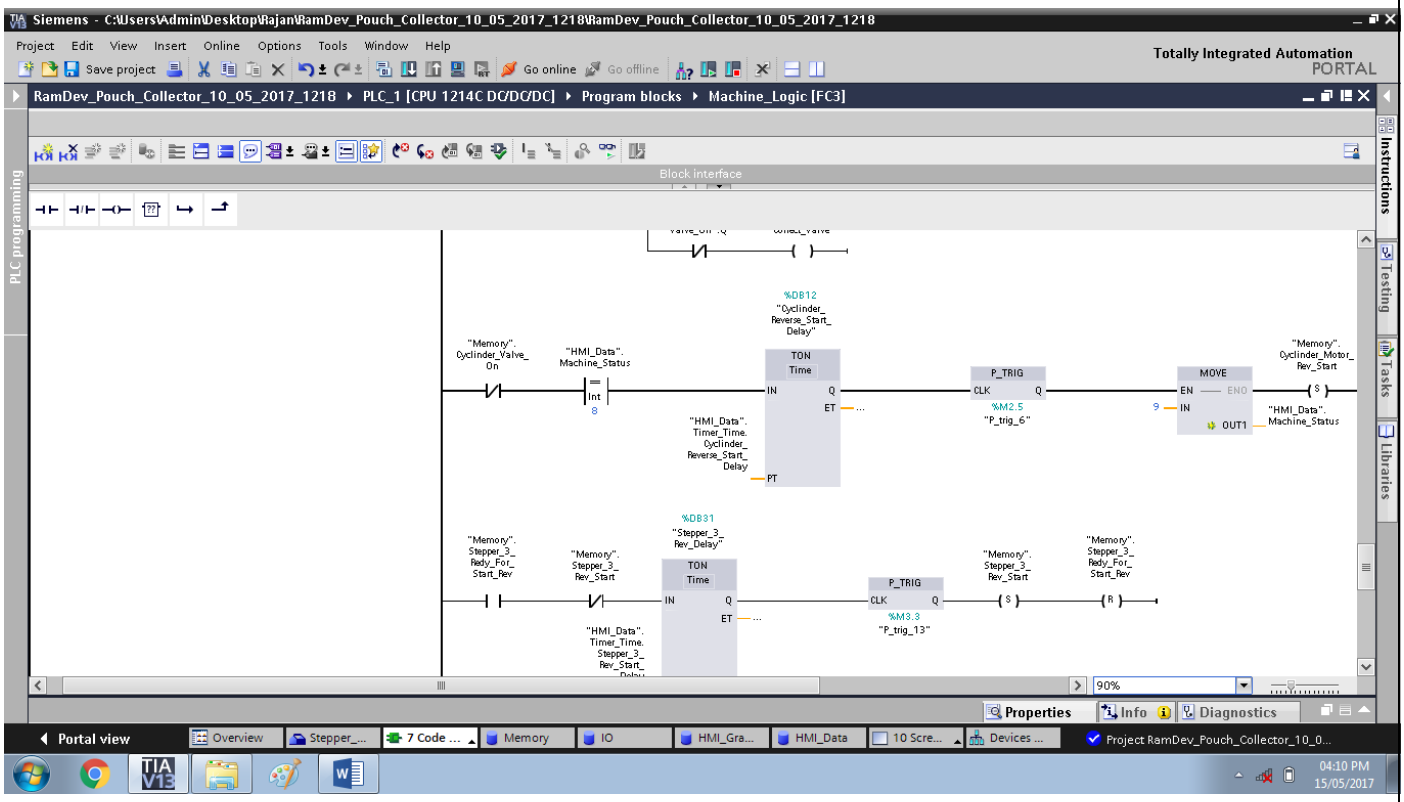

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
"IO".di.Stop_Proxy
Memory .belt_ Start
Time
TON
IN
Q
ET
PT
"HMI_Data". Timer_Time. Belt_Stop_ Delay
P_TRIG
CLK
%M2.1
"P_trig_2"
MOVE
EN
ENO
2
IN
OUT1
"HMI_Data". Machine_ Status_0
Network 3: Pouch Push // Valve-1
Machine Status = 3 = Pouch Push Running...
"HMI_Data". Auto_Mode_ Enable
"Memory". Machine_Power_ On
"IO".di.Start_Stop
"Memory".Belt_ Start
"IO".di.Stop_Proxy
"Memory". Pouch_Push
"Memory". Cyclinder_Motor_ Fwv_Start
"Memory". Cyclinder_Motor_ Rev_Start
"Memory". Cyclinder_Valve_ On
"HMI_Data". Machine_ Status_0
Int
2
%DB17
"Push_Mode_ Start_Safty_ Delay"
TON
Time
T#1s
%DB7
"Push_Delay"
%DB8
"Pouch_Push_ On_Timer"
90%
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:07 PM
15/05/2017

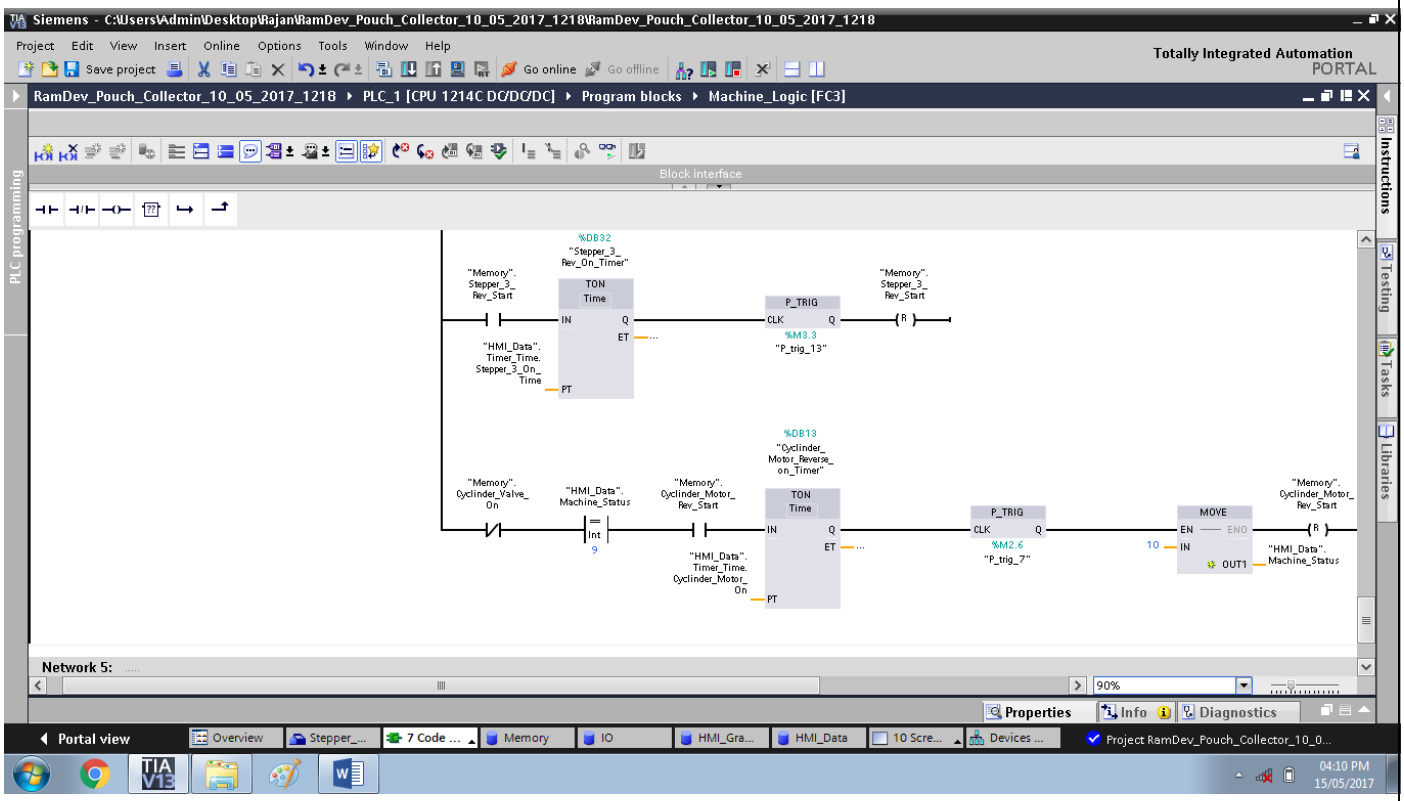

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
"Memory". Pouch_Push
"HMI_Data". Machine_Status
Int
3
%DB7
"Push_Delay"
TON
Time
IN
Q
ET
PT
"HMI_Data". Timer_Time. Pouch_Push_ Delay
%DB8
"Pouch_Push_ On_Timer"
"HMI_Data". Timer_Time. Pouch_push_ On
"Push_Delay".Q
"Pouch_Push_ On_Timer".Q
"IO".do.Strip_ Push_Valve
MOVE
EN
ENO
4
IN
OUT1
"Memory". Pouch_Push
"HMI_Data". Machine_Status
Network 4: Cyclinder Start/Stop // Stepper_2 + Stepper_3 & Valve-2
90%
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:07 PM
15/05/2017


Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
Network 4: Cyclinder Start/Stop // Stepper_2 + Stepper_3 & Valve-2
Machine Status = 5 = Cyclinder Motor Start Forward...
"HMI_Data". Auto_Mode_ Enable
"Memory". Machine_Power_ On
"IO".di.Start_Stop
"Memory". Pouch_Push
"HMI_Data". Machine_Status
%DB9 "Seq_2_Start_ Delay"
TON Time
"HMI_Data". Timer_Time. Seq_2_Start_ Delay
P_TRIG
%M2.3 "P_trig_4"
MOVE
"Memory". Cyclinder_Motor_ Frw_Start
"Memory". Stepper_3_ Redy_For_ Start_Frw
%DB29 "Stepper_3_ Frw_Start_ Delay"
"Memory". Stepper_3_ Frw_Start
%M3.1 "P_trig_11"
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:08 PM 15/05/2017

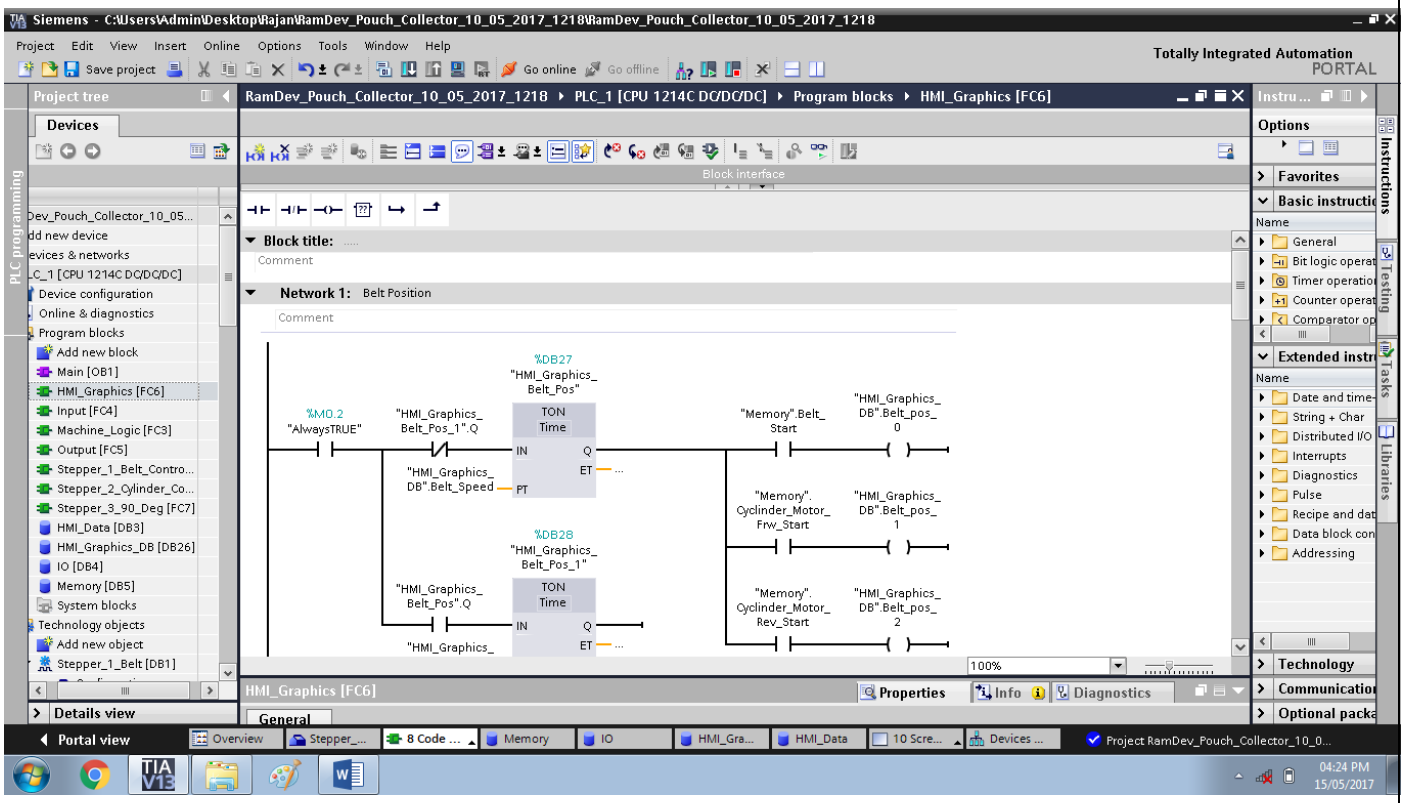

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
%DB30 "Stepper_3_ On_Timer"
"Memory". Stepper_3_ Frw_Start
TON Time
"HMI_Data". Timer_Time. Stepper_3_On_ Time
P_TRIG
%M3.2 "P_trig_12"
%DB10 "Cyclinder_ Motor_Frw_On_ Timer"
"Memory". Cyclinder_Motor_ Frw_Start
"HMI_Data". Machine_Status
"HMI_Data". Timer_Time. Cyclinder_Motor_ On
MOVE
"HMI_Data". Machine_Status
"Memory". Cyclinder_Valve_ On
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:09 PM 15/05/2017

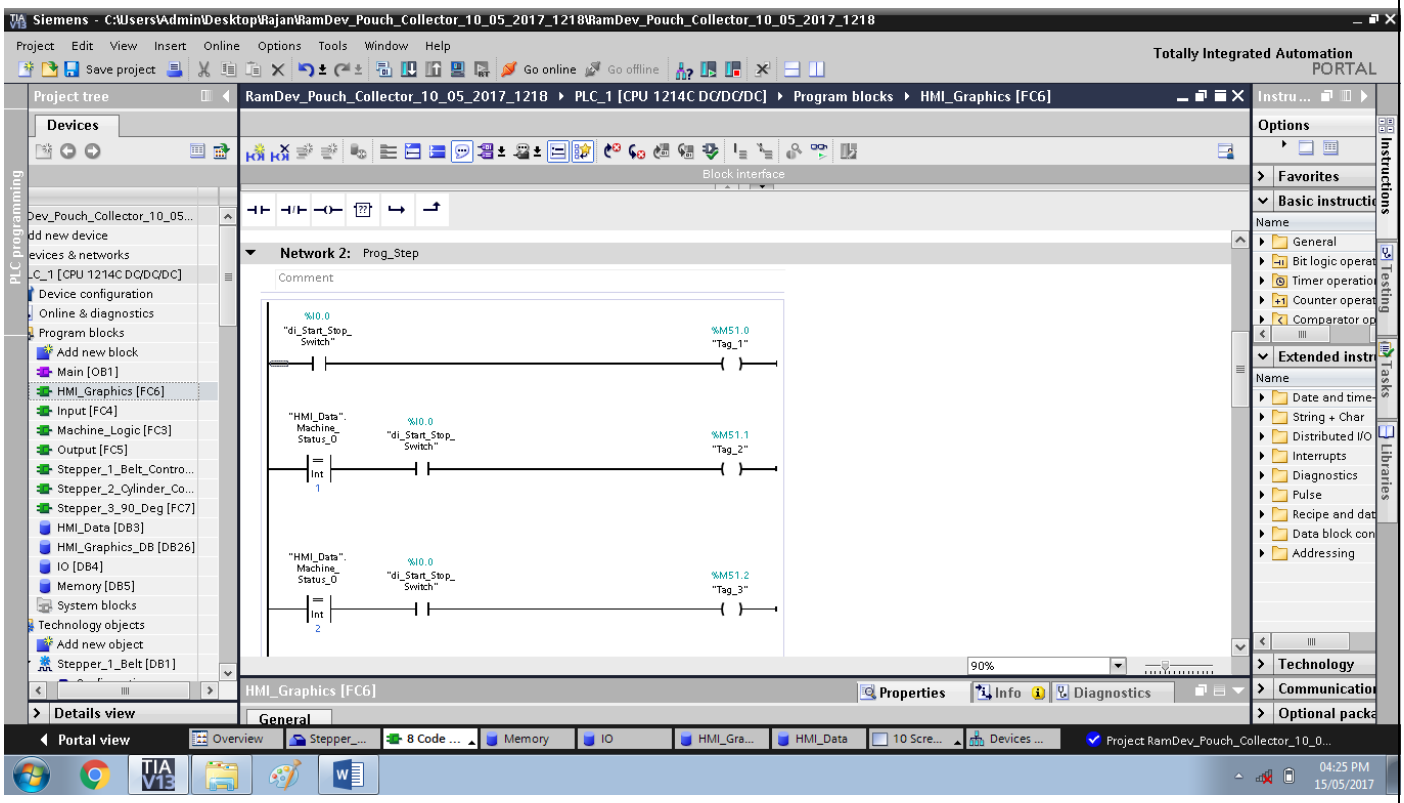

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
PLC programming
Instructions
Testing
Tasks
Libraries
"Memory". Cyclinder_Motor_ Fwv_Start
"HMI_Data". Machine_Status
P_TRIG
%M2.4
"P_trig_5"
MOVE
"Memory". Cyclinder_Valve_ On
"HMI_Data". Machine_Status
%DB11
"Cyclinder_ Valve_On"
TON
Time
"HMI_Data". Timer_Time. Cyclinder_Valve_ On
"Cyclinder_ Valve_On".Q
"IO".do.Strip_ Collect_Valve
%DB12
90%
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:09 PM
15/05/2017

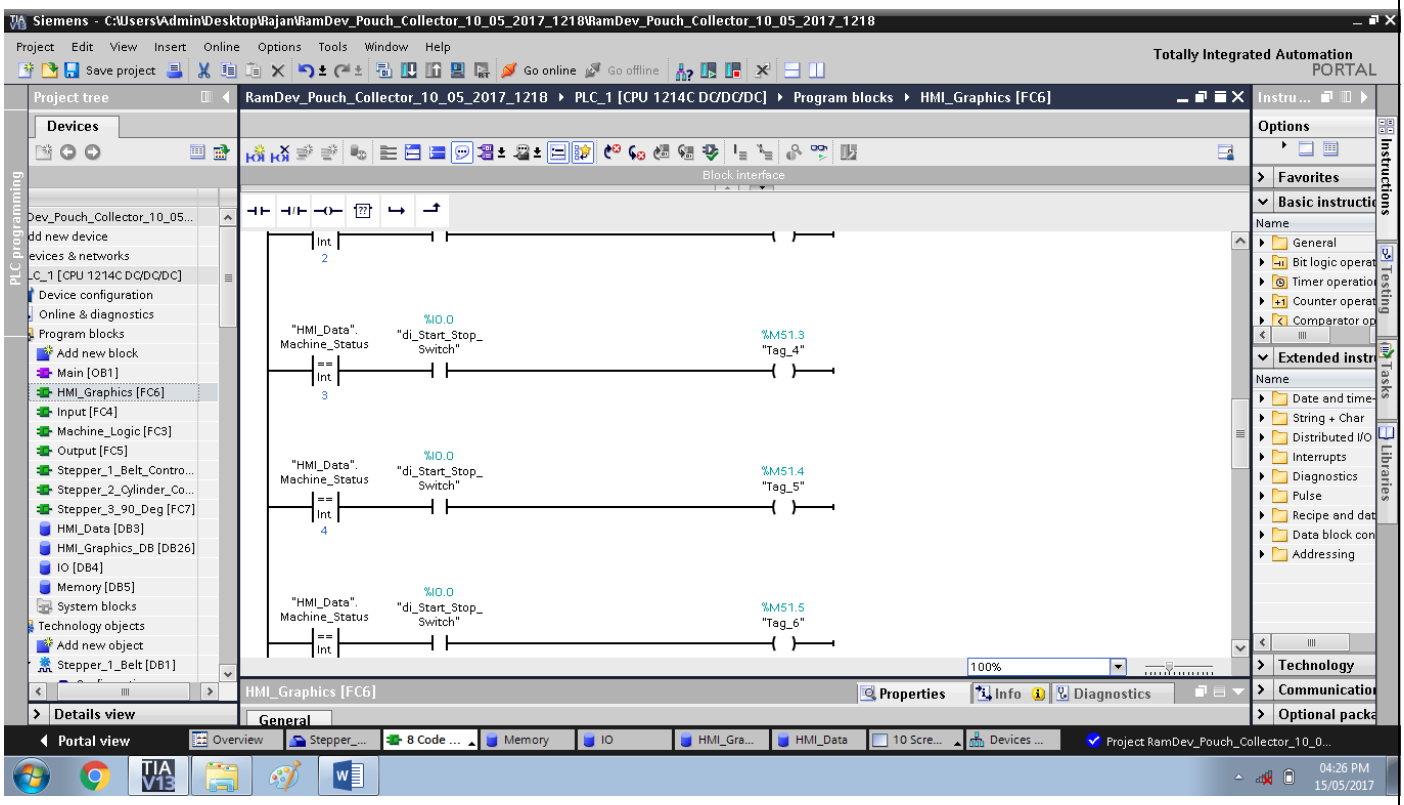

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ Machine_Logic [FC3]
Block interface
PLC programming
Instructions
Testing
Tasks
Libraries
%DB12
"Cyclinder_ Reverse_Start_ Delay"
"Memory". Cyclinder_Valve_ On
"HMI_Data". Machine_Status
TON
Time
"HMI_Data". Timer_Time. Cyclinder_ Reverse_Start_ Delay
P_TRIG
%M2.5
"P_trig_6"
MOVE
"Memory". Cyclinder_Motor_ Rev_Start
"HMI_Data". Machine_Status
%DB31
"Stepper_3_ Rev_Delay"
"Memory". Stepper_3_ Redy_For_ Start_Rev
"Memory". Stepper_3_ Rev_Start
"HMI_Data". Timer_Time. Stepper_3_ Rev_Start_
%M3.3
"P_trig_13"
90%
Properties
Info
Diagnostics
Portal view
Overview
Stepper_...
7 Code ...
Memory
IO
HMI_Gra...
HMI_Data
10 Scre...
Devices ...
Project RamDev_Pouch_Collector_10_0...
04:10 PM
15/05/2017

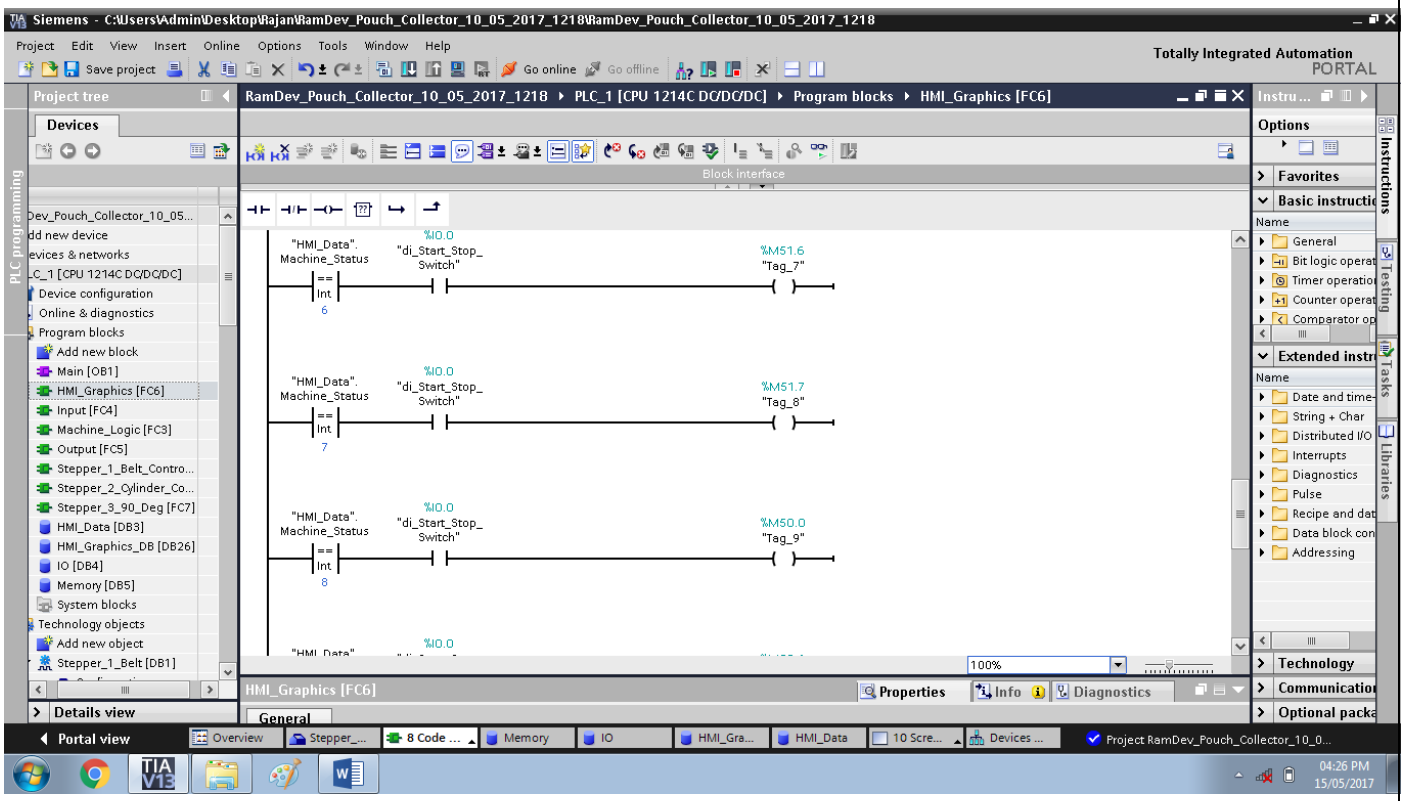


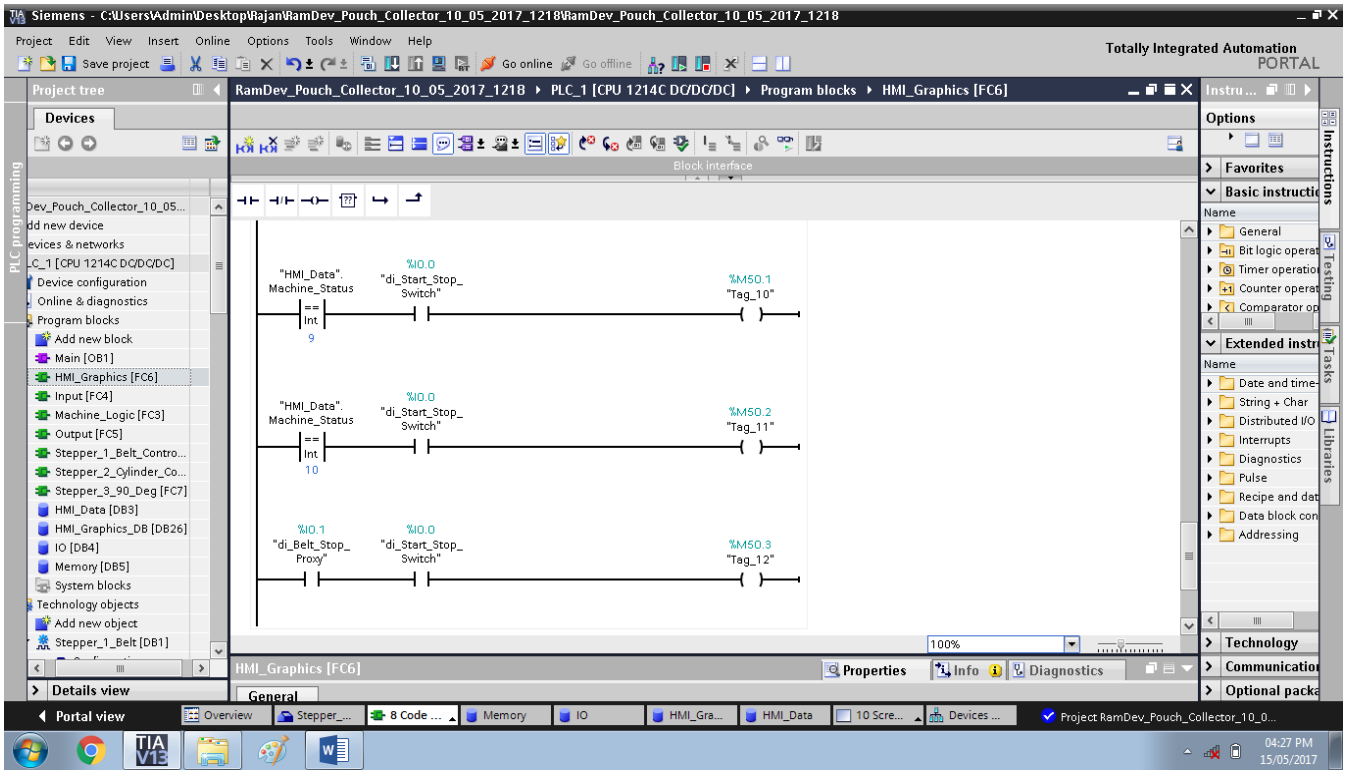
8). HMI Graphics.



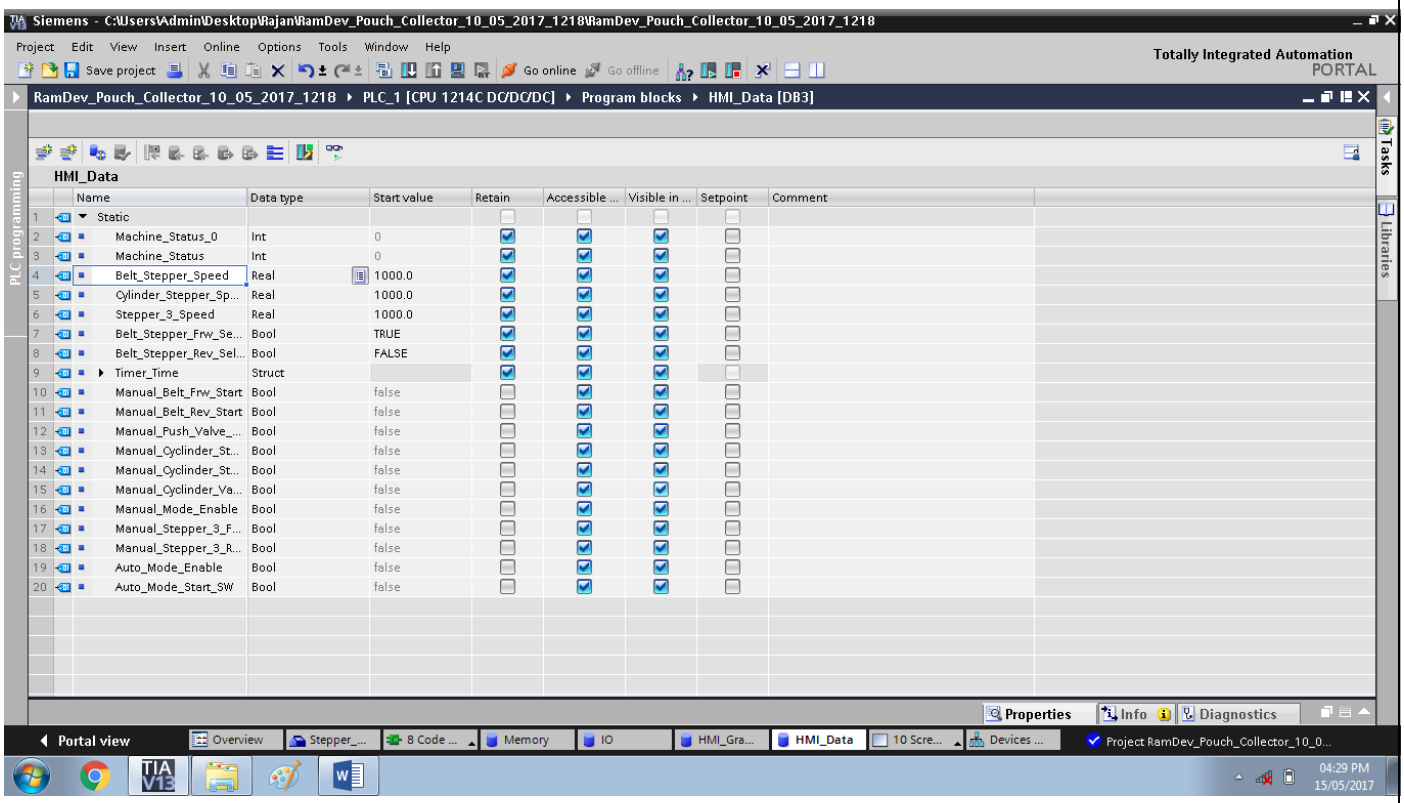

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ HMI_Graphics [FC6]
Block interface
Network 2: Prog_Step
Comment
%I0.0 "di_Start_Stop_Switch"
%M51.0 "Tag_1"
"HMI_Data".Machine_Status_0 == Int 1
%I0.0 "di_Start_Stop_Switch"
%M51.1 "Tag_2"
"HMI_Data".Machine_Status_0 == Int 2
%I0.0 "di_Start_Stop_Switch"
%M51.2 "Tag_3"
Add new device
Add new block
Main [OB1]
HMI_Graphics [FC6]
Input [FC4]
Machine_Logic [FC3]
Output [FC5]
Stepper_1_Belt_Contro...
Stepper_2_Cylinder_Co...
Stepper_3_90_Deg [FC7]
HMI_Data [DB3]
HMI_Graphics_DB [DB26]
IO [DB4]
Memory [DB5]
System blocks
Technology objects
Add new object
Stepper_1_Belt [DB1]
Details view
Portal view
Overview
Properties
Info
Diagnostics
Favorites
Basic instructions
Extended instructions
Technology
Communication
Optional packages
04:25 PM
15/05/2017


Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ HMI_Graphics [FC6]
Block interface
Int 2
"HMI_Data".Machine_Status == Int 3
%I0.0 "di_Start_Stop_Switch"
%M51.3 "Tag_4"
"HMI_Data".Machine_Status == Int 4
%I0.0 "di_Start_Stop_Switch"
%M51.4 "Tag_5"
"HMI_Data".Machine_Status == Int
%I0.0 "di_Start_Stop_Switch"
%M51.5 "Tag_6"
100%
Portal view
Overview
Properties
Info
Diagnostics
04:26 PM
15/05/2017

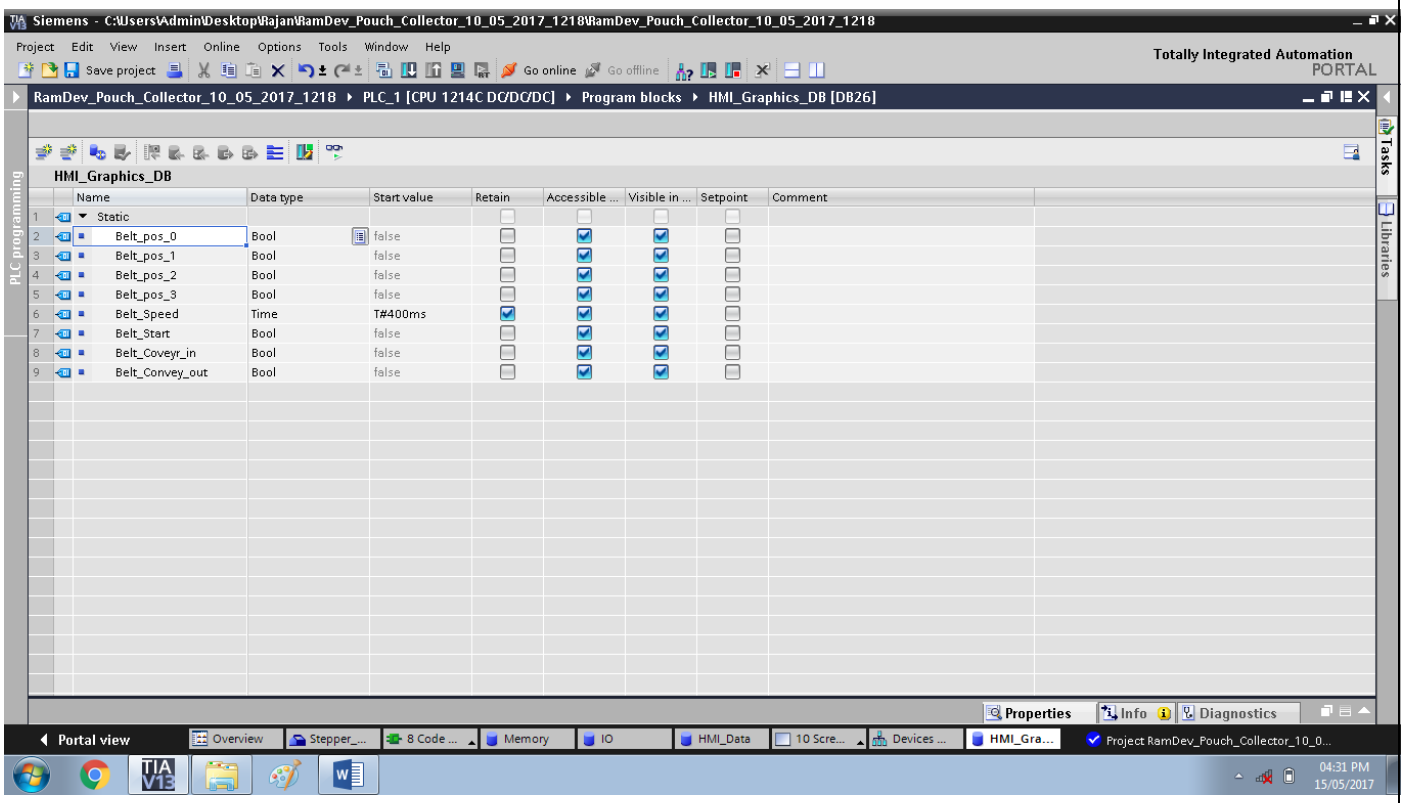

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ HMI_Graphics [FC6]
Block interface
"HMI_Data". Machine_Status == Int 6
%I0.0 "di_Start_Stop_ Switch"
%M51.6 "Tag_7"
"HMI_Data". Machine_Status == Int 7
%I0.0 "di_Start_Stop_ Switch"
%M51.7 "Tag_8"
"HMI_Data". Machine_Status == Int 8
%I0.0 "di_Start_Stop_ Switch"
%M50.0 "Tag_9"
Add new block
Main [OB1]
HMI_Graphics [FC6]
Input [FC4]
Machine_Logic [FC3]
Output [FC5]
HMI_Data [DB3]
HMI_Graphics_DB [DB26]
IO [DB4]
Memory [DB5]
System blocks
Technology objects
Add new object
Stepper_1_Belt [DB1]
Details view
Portal view
Overview
HMI_Graphics [FC6]
Properties
Info
Diagnostics
General
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
Project RamDev_Pouch_Collector_10_0...
04:26 PM
15/05/2017


Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
Project tree
Devices
RamDev_Pouch_Collector_10_05_2017_1218 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ HMI_Graphics [FC6]
Block interface
"HMI_Data". Machine_Status == Int 9
%I0.0 "di_Start_Stop_ Switch"
%M50.1 "Tag_10"
"HMI_Data". Machine_Status == Int 10
%I0.0 "di_Start_Stop_ Switch"
%M50.2 "Tag_11"
%I0.1 "di_Belt_Stop_ Proxy"
%I0.0 "di_Start_Stop_ Switch"
%M50.3 "Tag_12"
Add new block
Main [OB1]
HMI_Graphics [FC6]
Input [FC4]
Machine_Logic [FC3]
Output [FC5]
HMI_Data [DB3]
HMI_Graphics_DB [DB26]
IO [DB4]
Memory [DB5]
System blocks
Technology objects
Add new object
Stepper_1_Belt [DB1]
Details view
Portal view
Overview
HMI_Graphics [FC6]
Properties
Info
Diagnostics
General
Options
Favorites
Basic instructi
Extended instr
Technology
Communicatio
Optional packa
Project RamDev_Pouch_Collector_10_0...
04:27 PM
15/05/2017

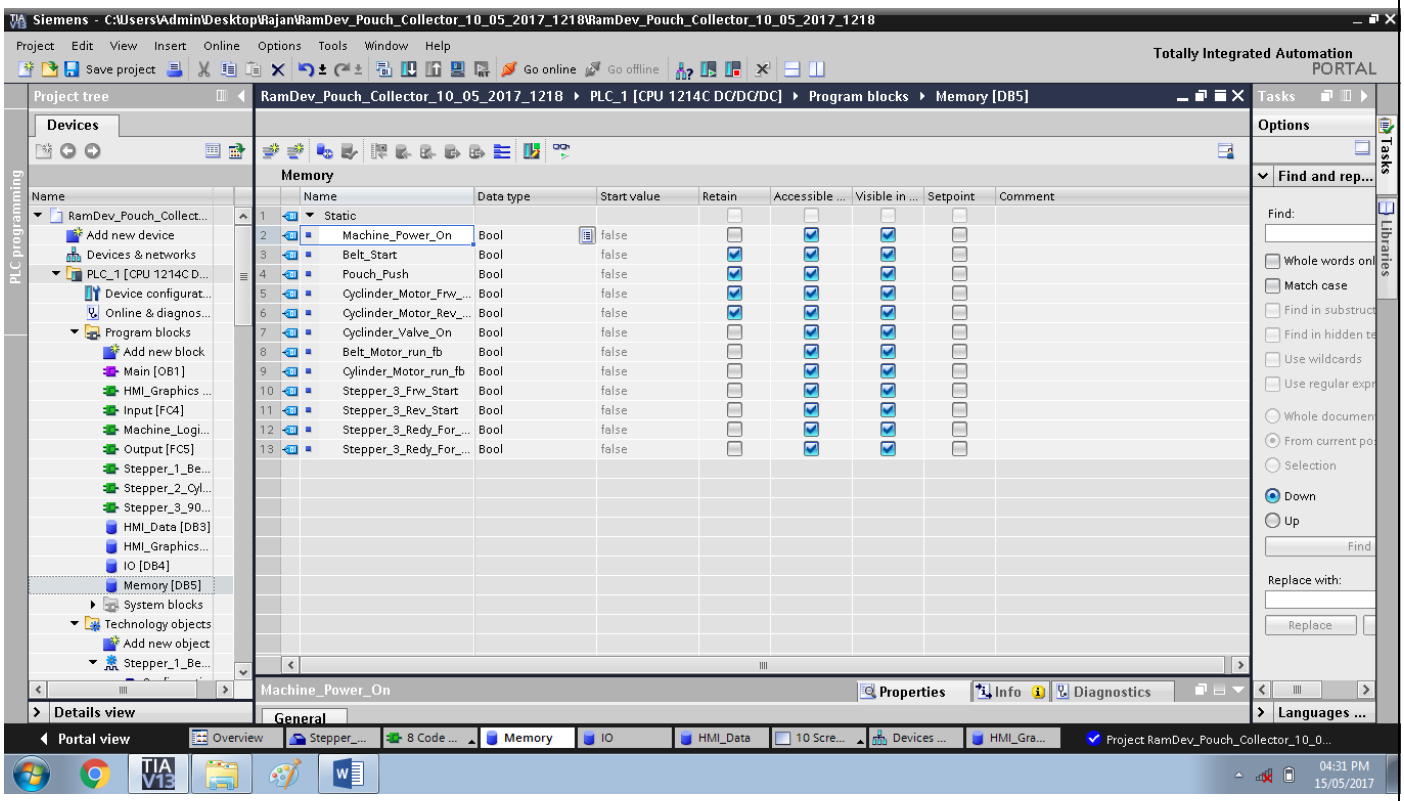
9). HMI Data Types



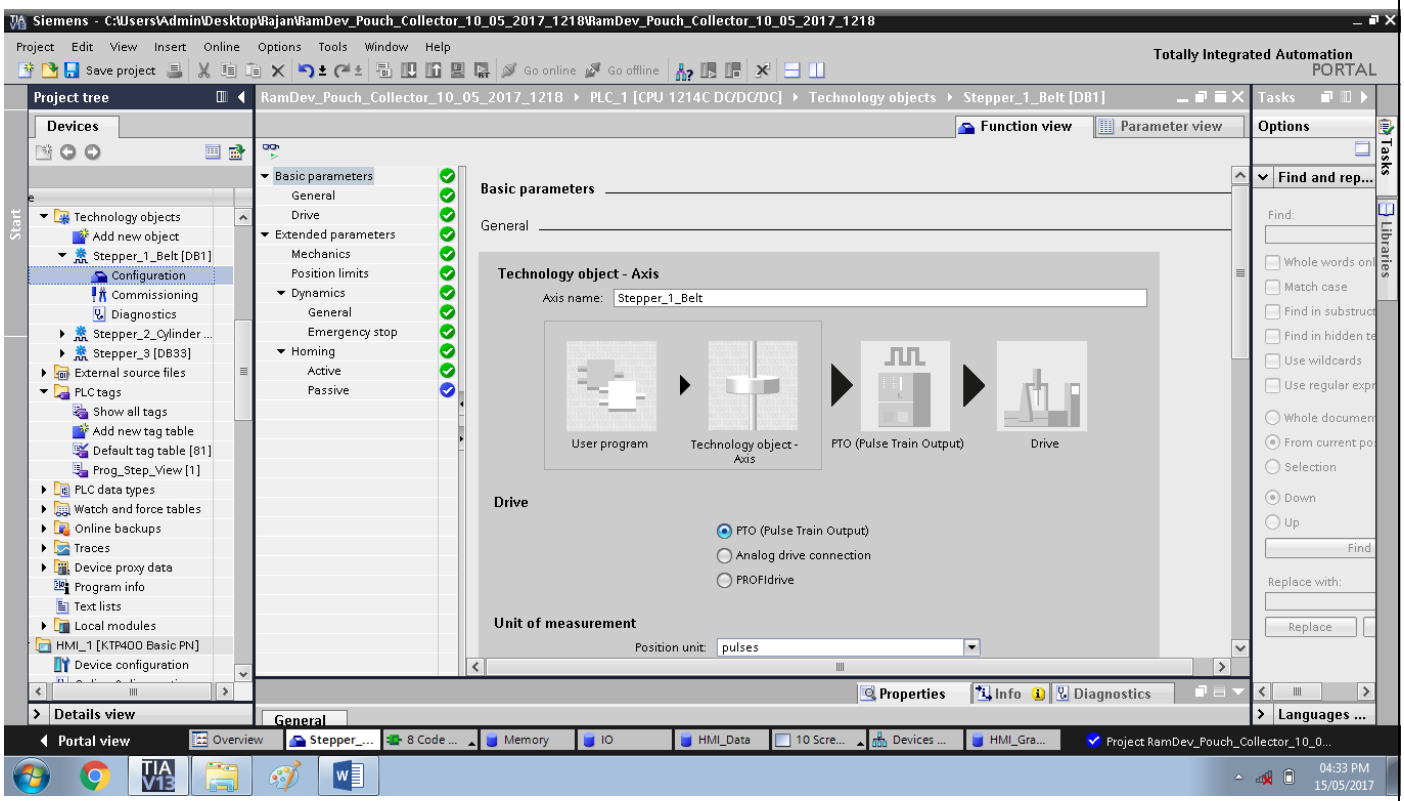
10). HMI Graphics Data Type/Block



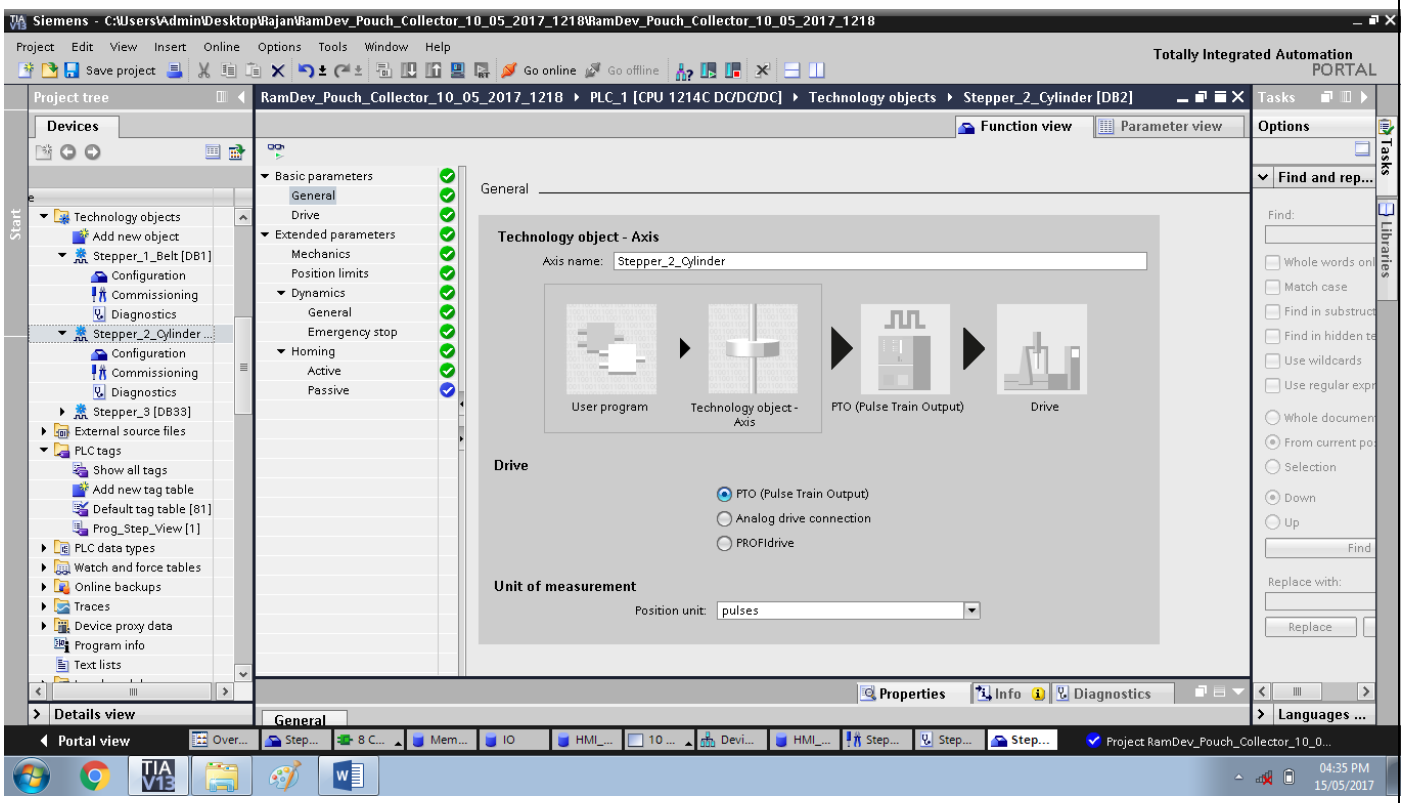
11). Memory DB



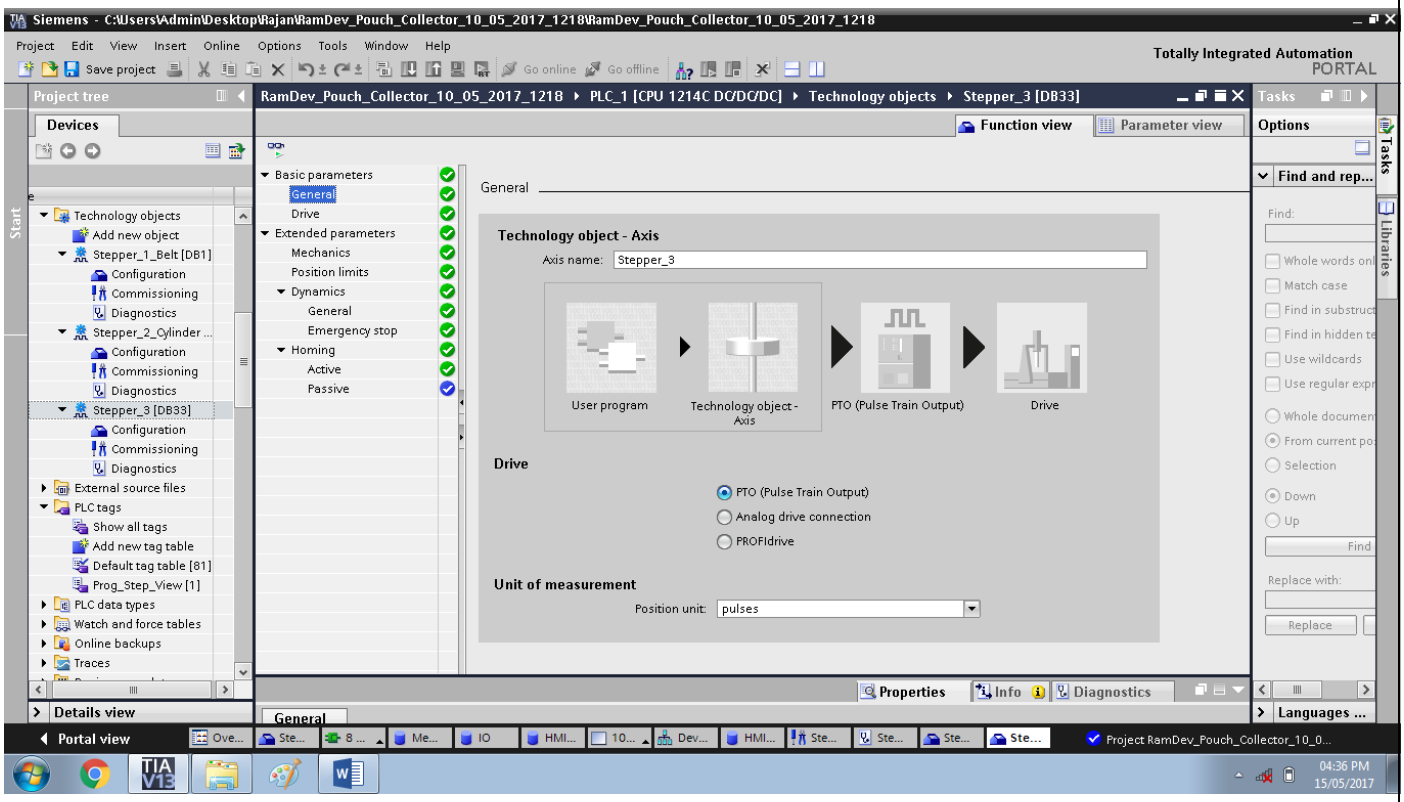
12). Stepper-1 Configuration



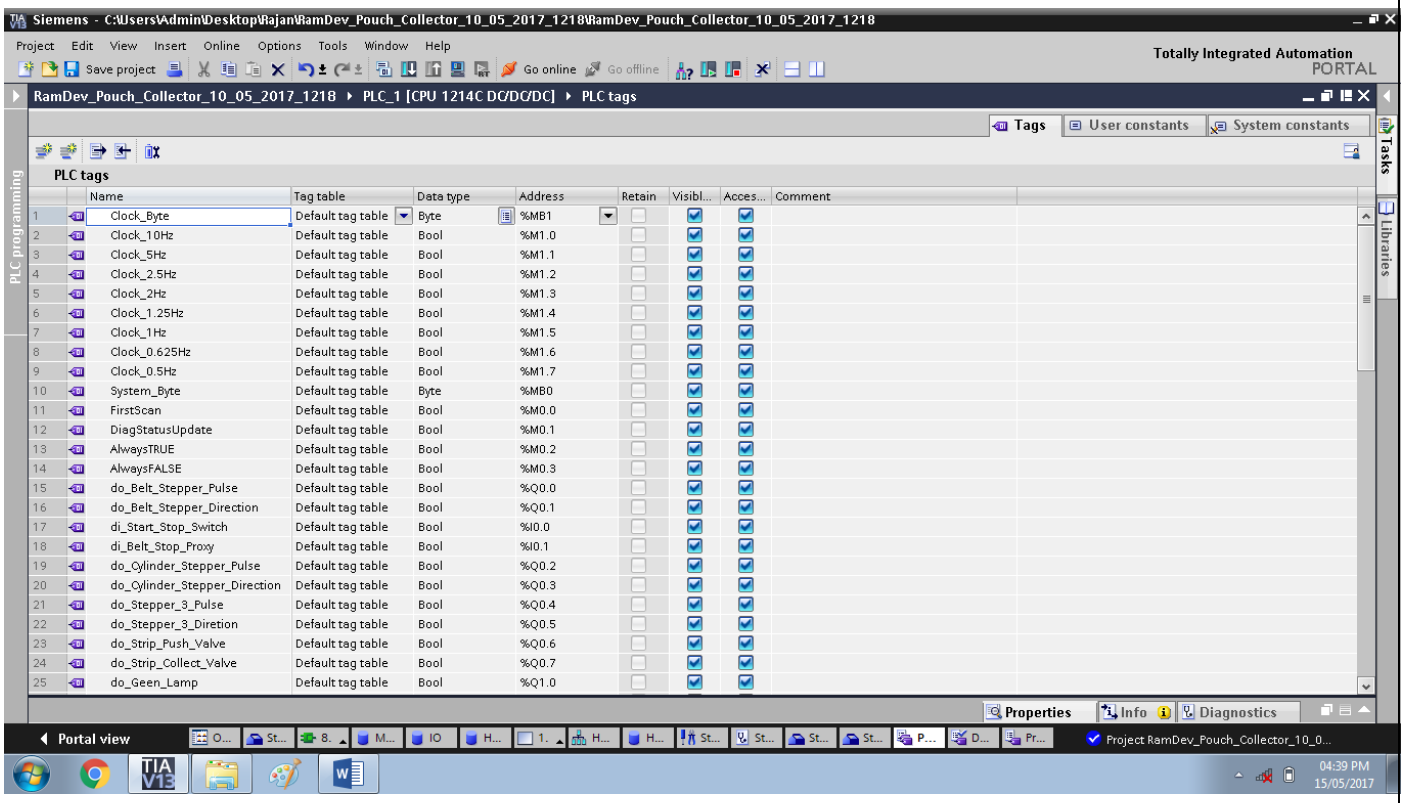
13). Stepper-2 Configuration

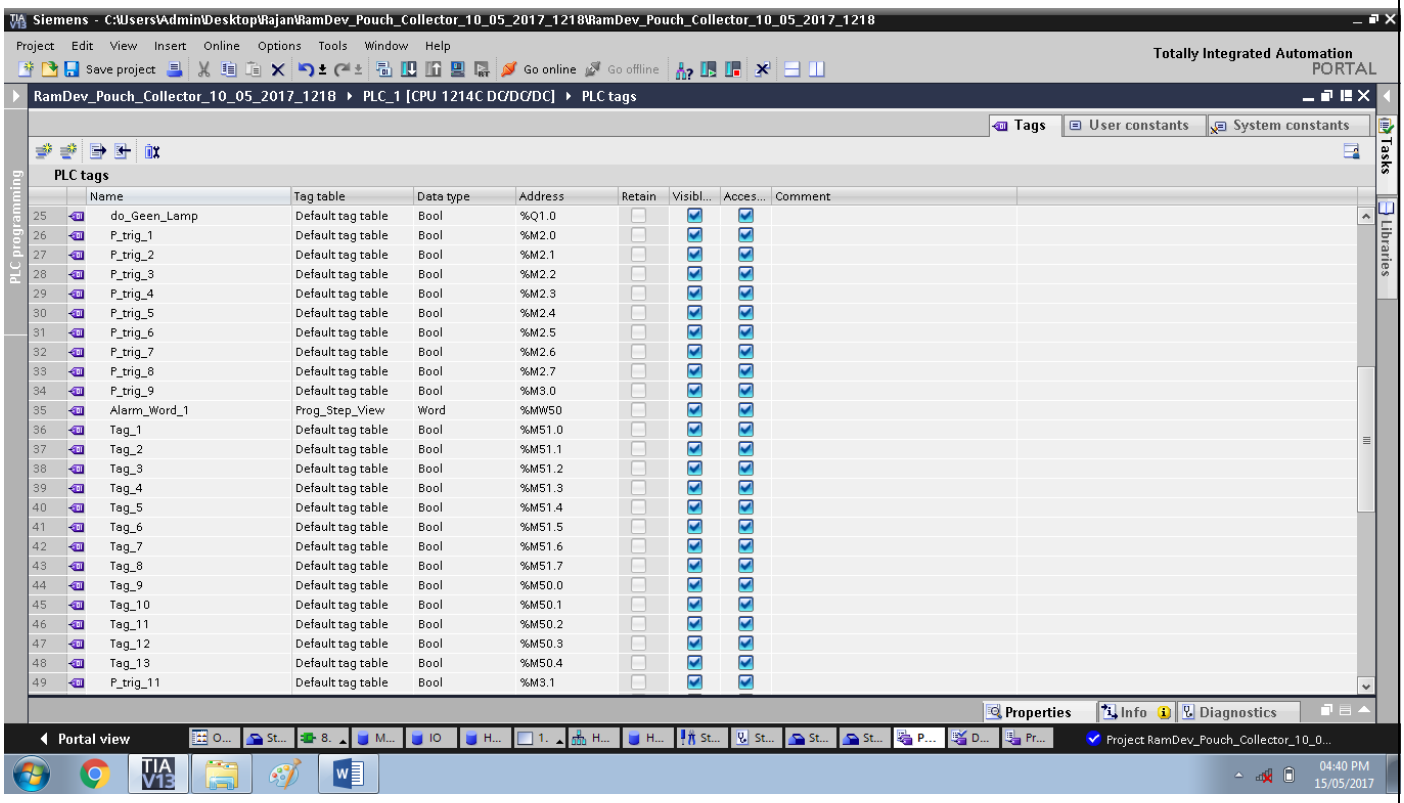


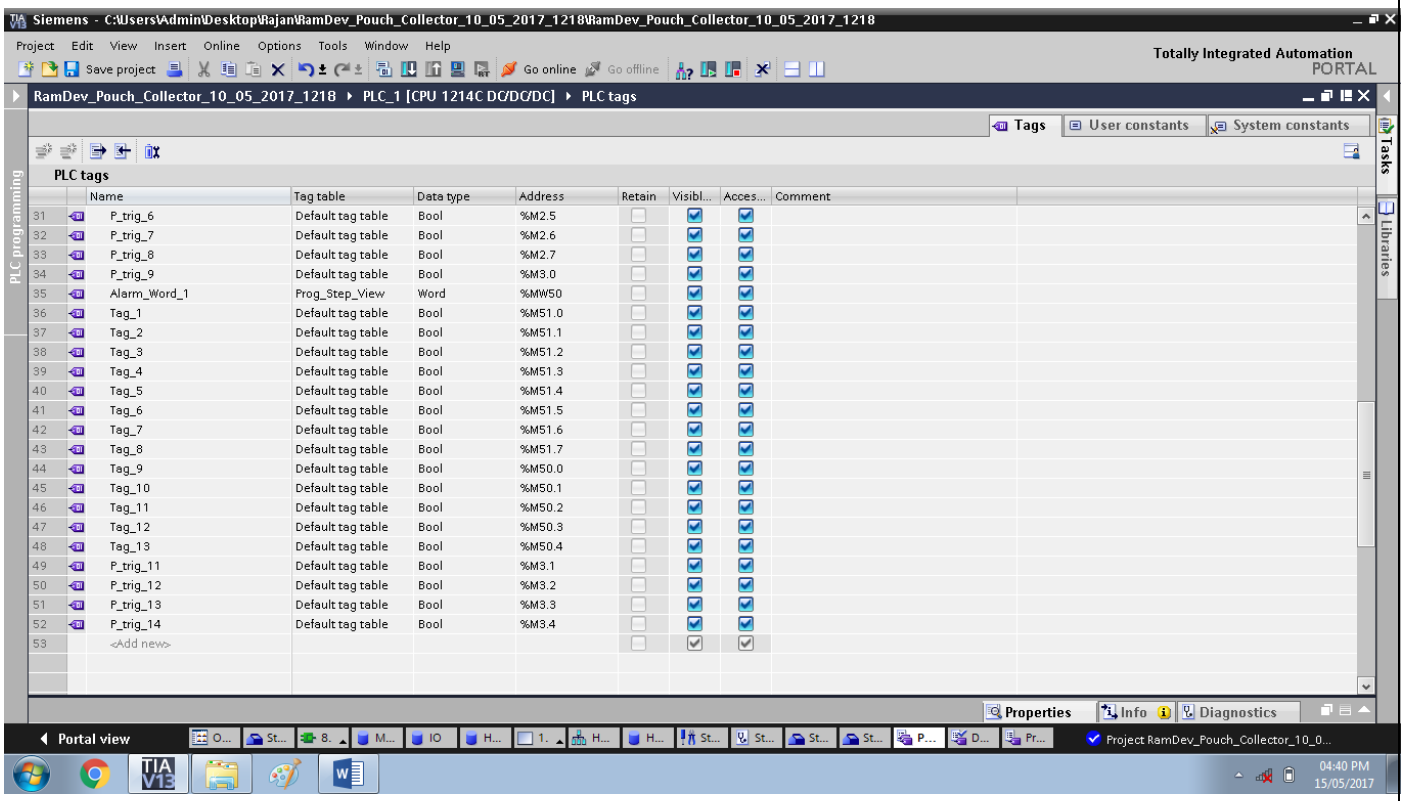
14). Stepper-3 Configuration



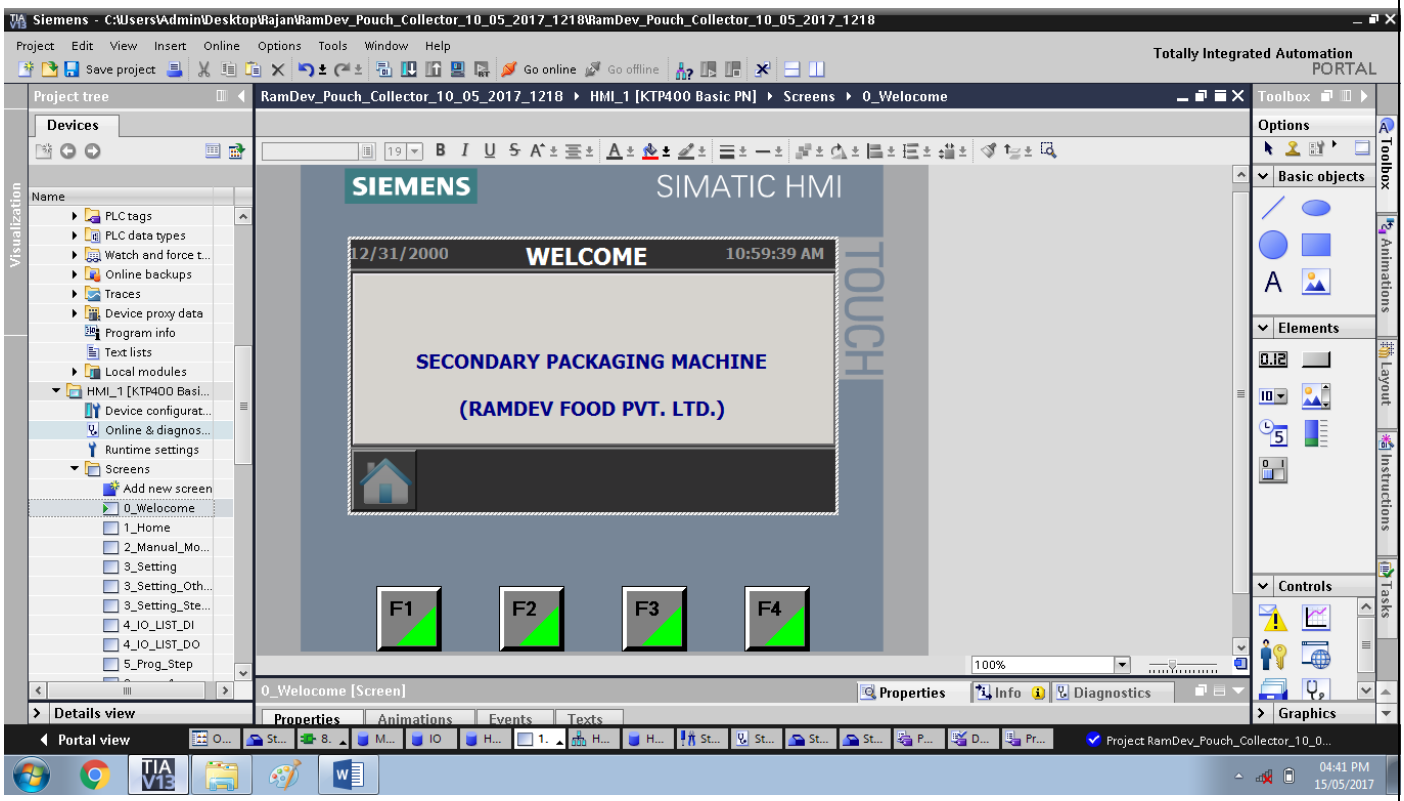
15). Tag Table/ Mapping

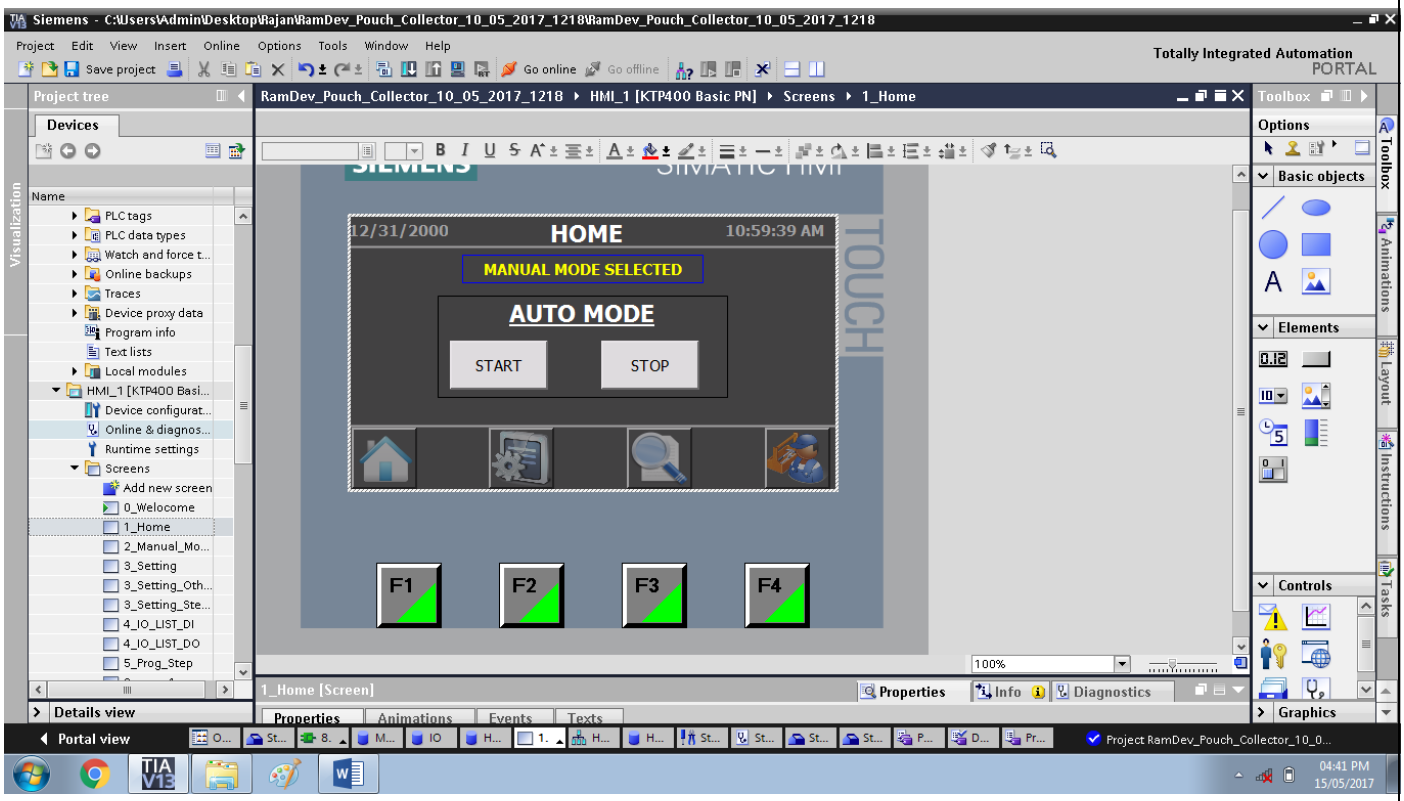


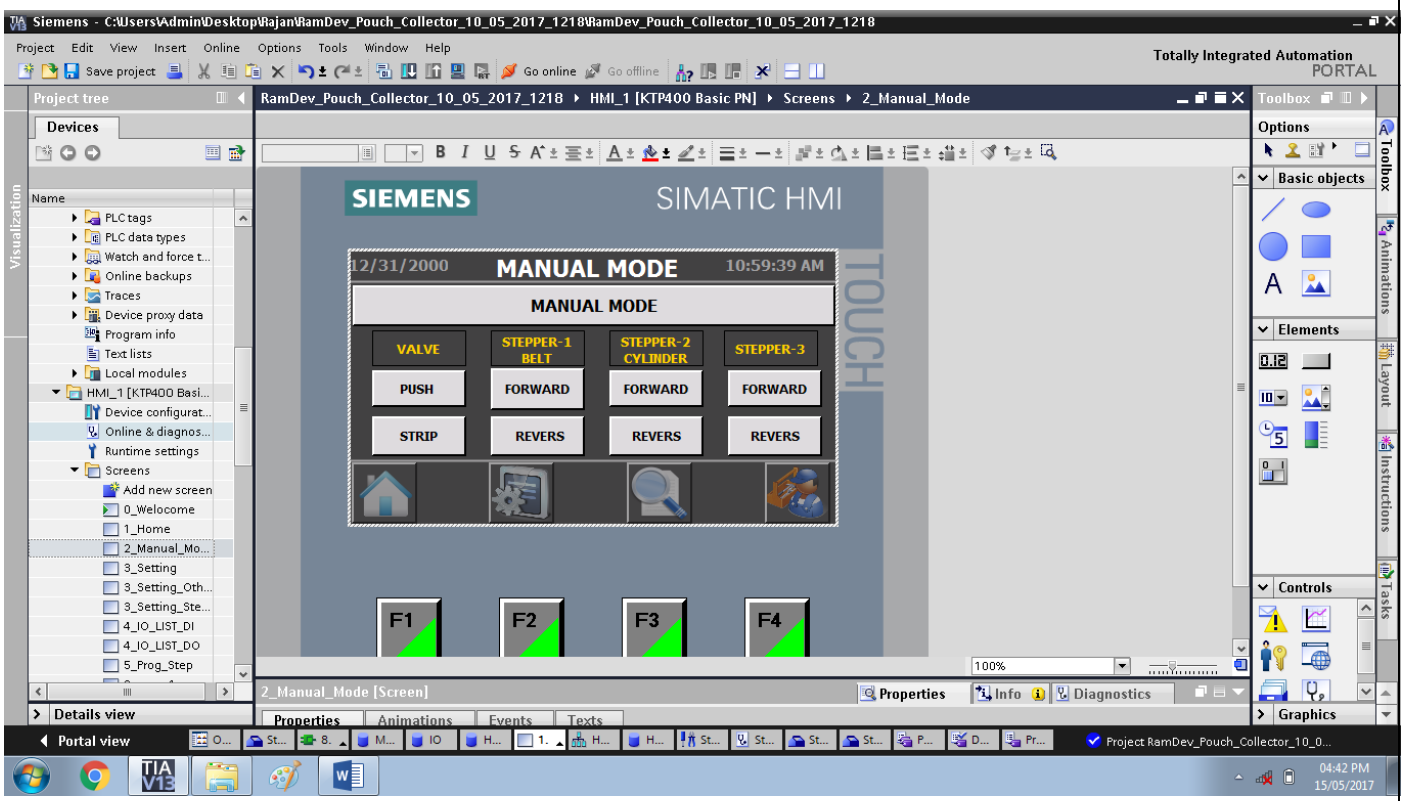


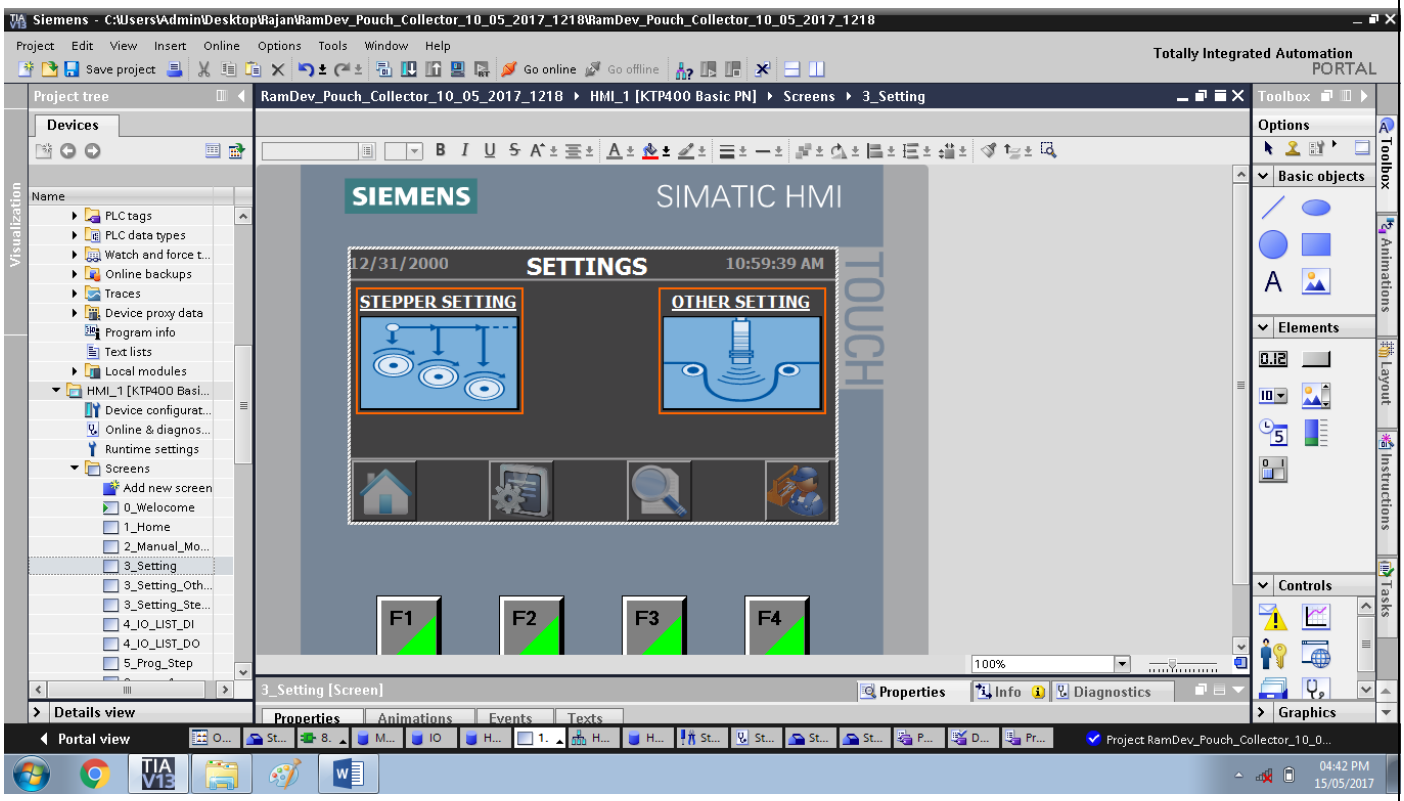


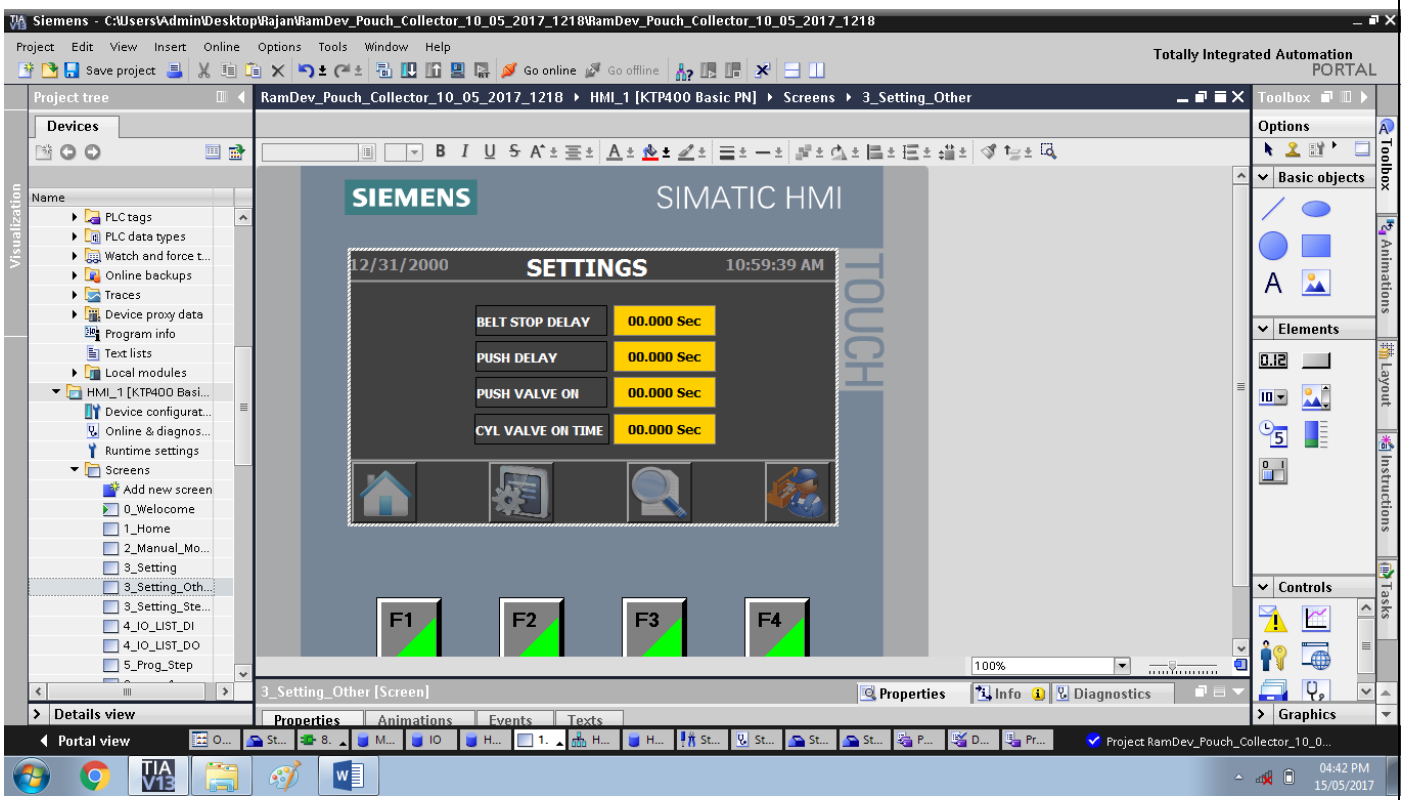
16). HMI Screens

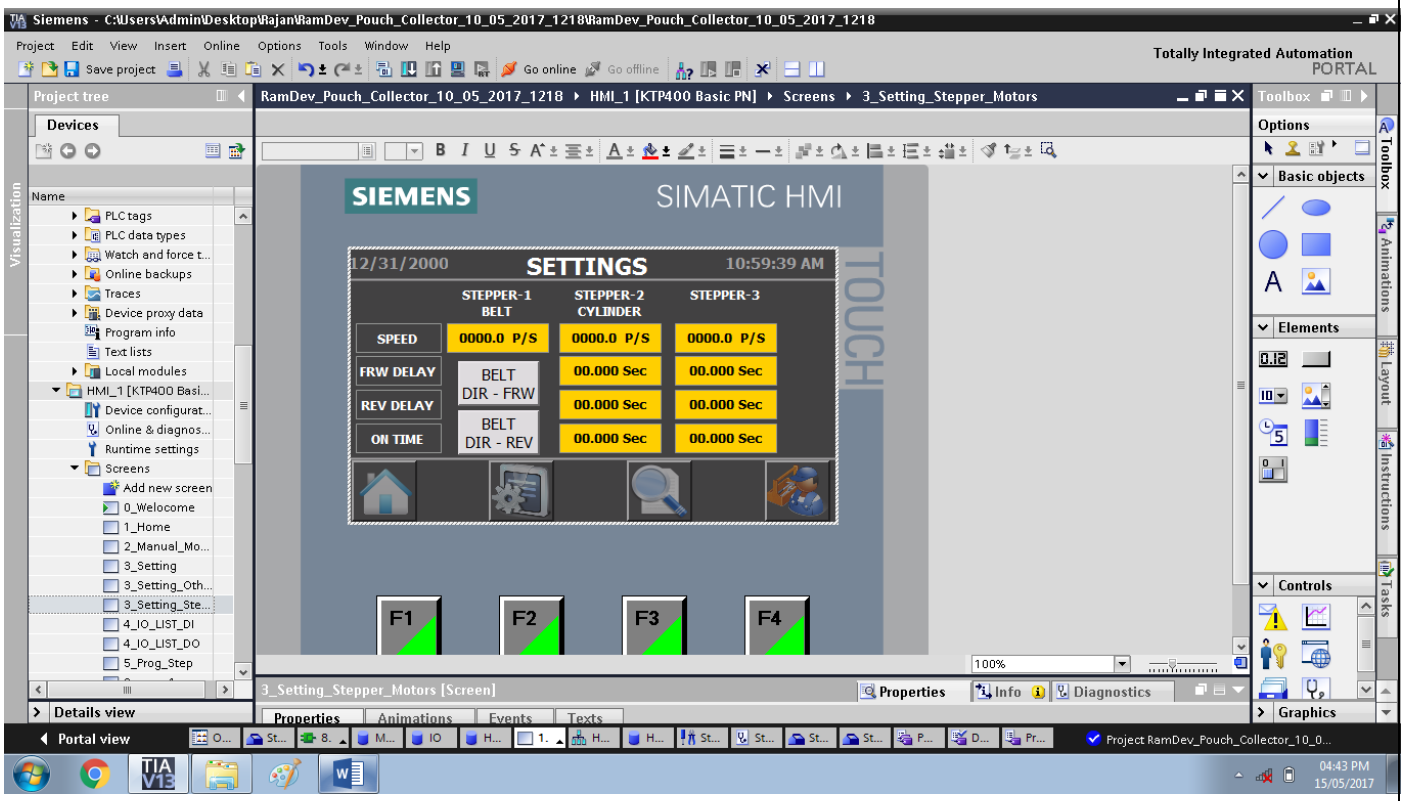


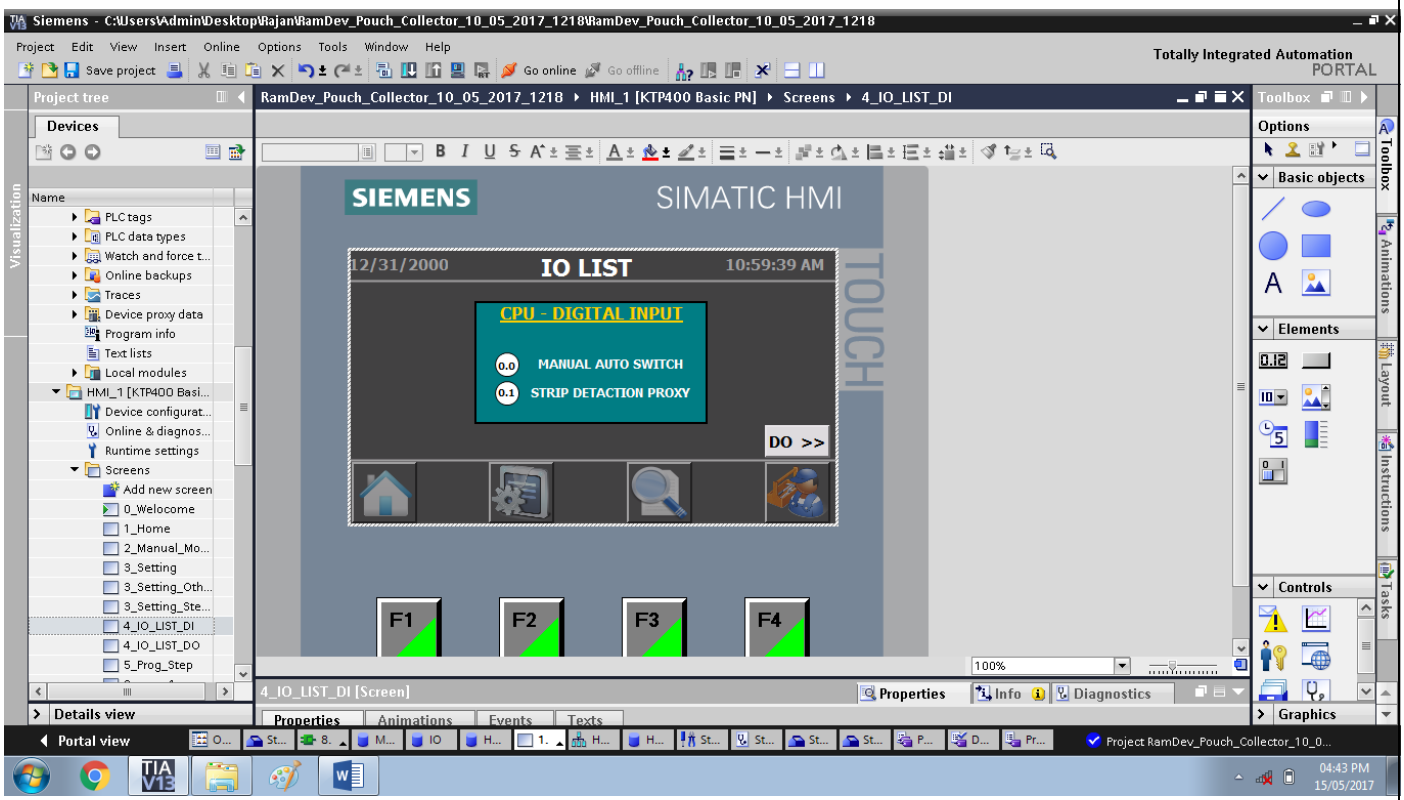

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 1_Home
SIEMENS
SIMATIC HMI
12/31/2000
HOME
10:59:39 AM
MANUAL MODE SELECTED
AUTO MODE
START
STOP
TOUCH
F1
F2
F3
F4
1_Home [Screen]

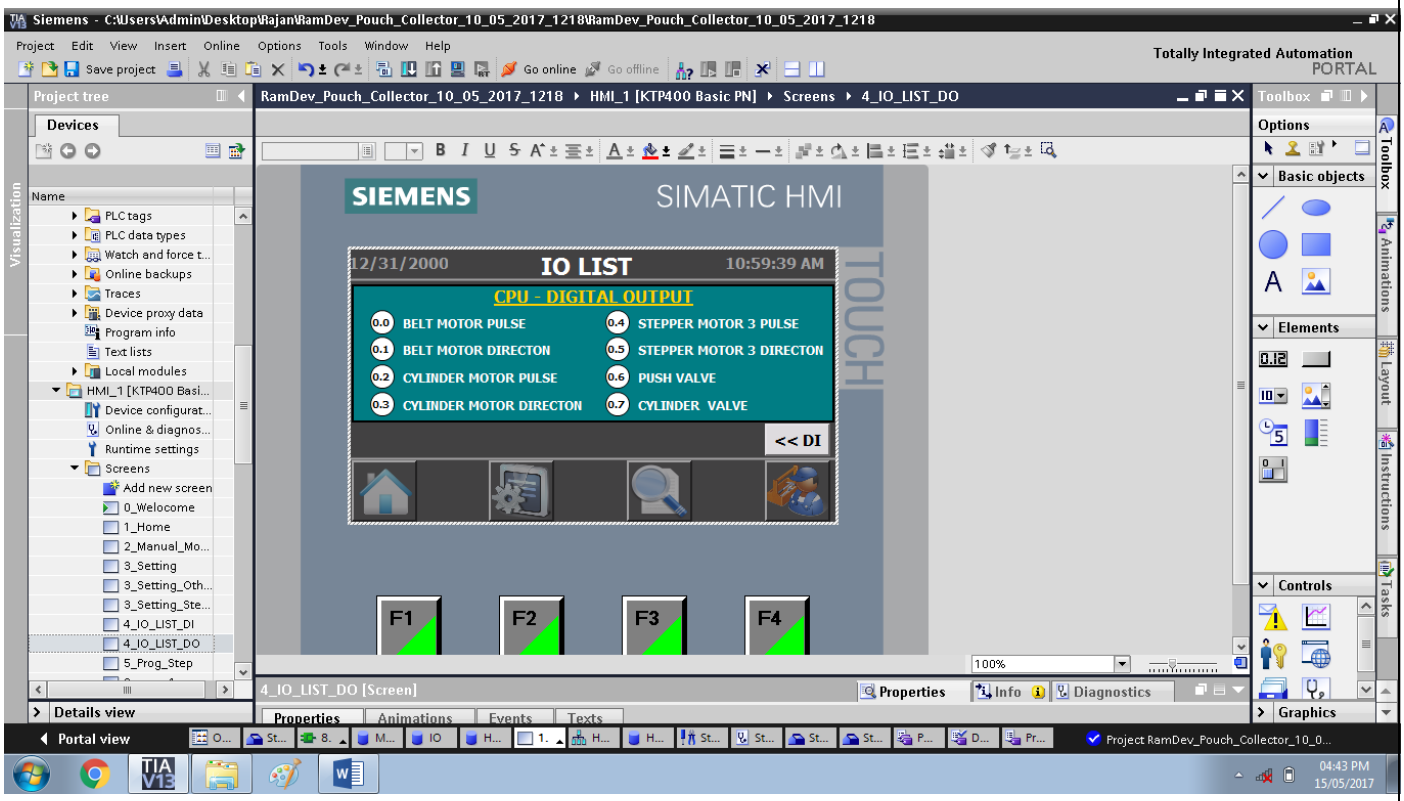

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 2_Manual_Mode
SIEMENS
SIMATIC HMI
12/31/2000
MANUAL MODE
10:59:39 AM
MANUAL MODE
VALVE
STEPPER-1 BELT
STEPPER-2 CYLINDER
STEPPER-3
PUSH
FORWARD
FORWARD
FORWARD
STRIP
REVERS
REVERS
REVERS
TOUCH
F1
F2
F3
F4
2_Manual_Mode [Screen]

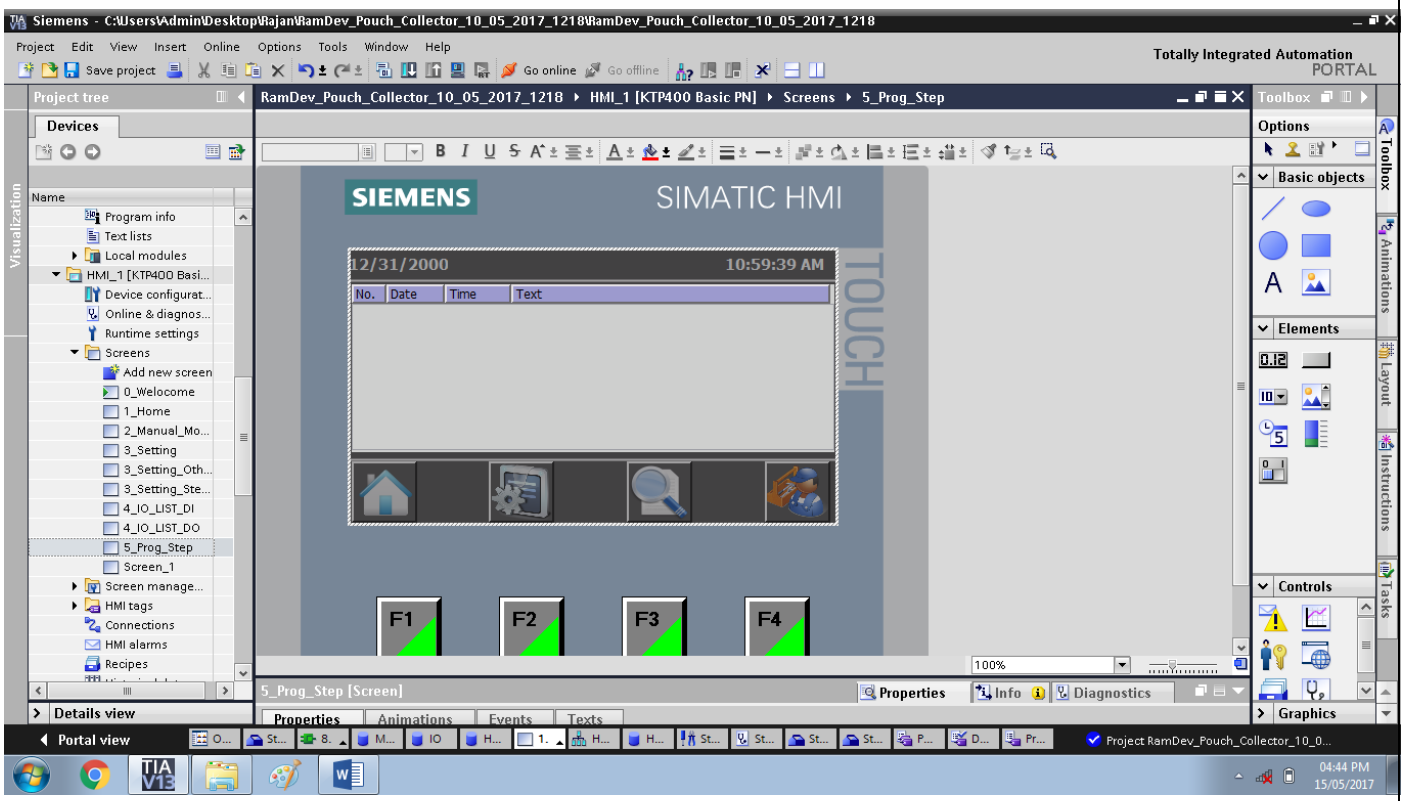

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 3_Setting
SIEMENS
SIMATIC HMI
12/31/2000
SETTINGS
10:59:39 AM
TOUCH
STEPPER SETTING
OTHER SETTING
F1
F2
F3
F4
3_Setting [Screen]


Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 3_Setting_Other
SIEMENS
SIMATIC HMI
12/31/2000
SETTINGS
10:59:39 AM
TOUCH
BELT STOP DELAY
00.000 Sec
PUSH DELAY
00.000 Sec
PUSH VALVE ON
00.000 Sec
CYL VALVE ON TIME
00.000 Sec
F1
F2
F3
F4
3_Setting_Other [Screen]

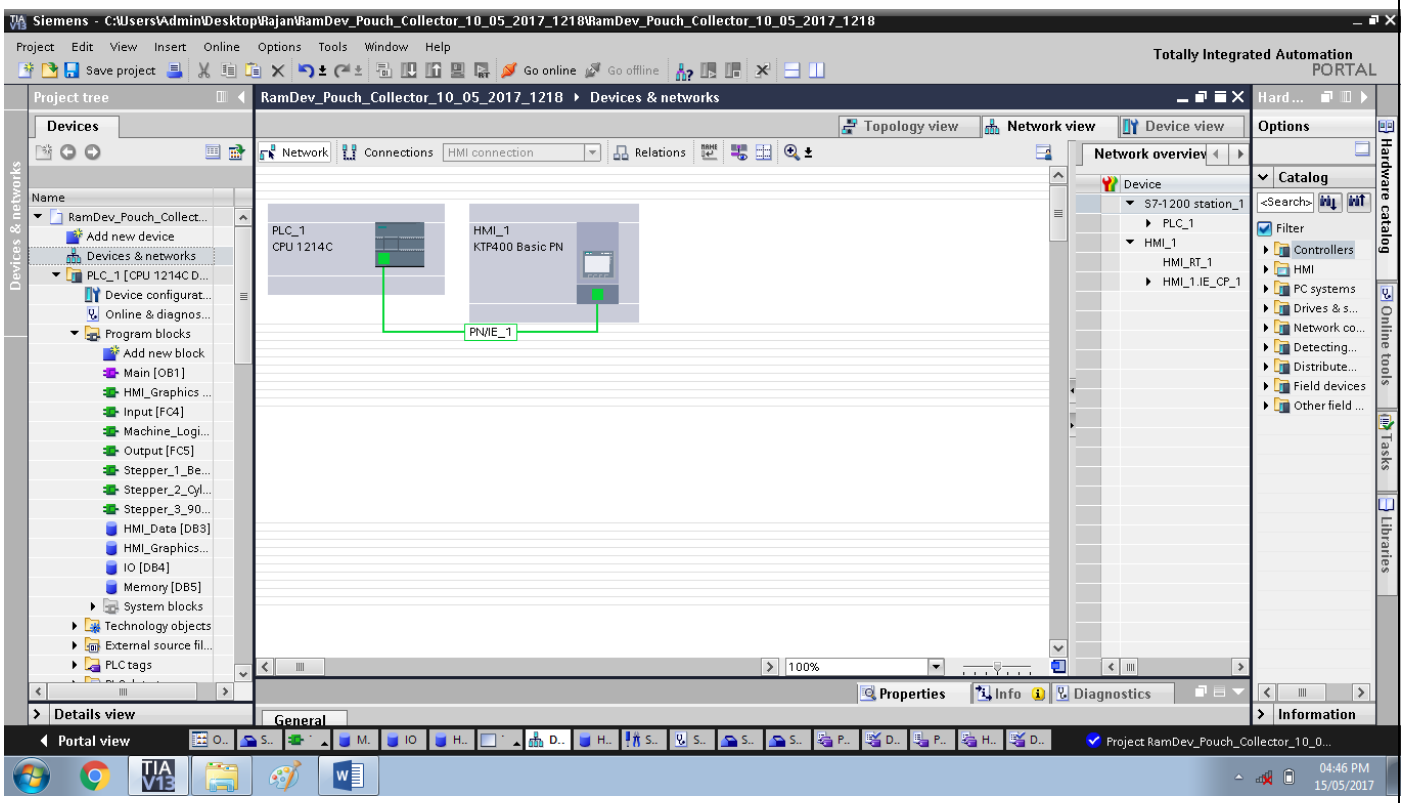

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 3_Setting_Stepper_Motors
SIEMENS
SIMATIC HMI
12/31/2000
SETTINGS
10:59:39 AM
TOUCH
STEPPER-1 BELT
STEPPER-2 CYLINDER
STEPPER-3
SPEED
0000.0 P/S
FRW DELAY
REV DELAY
ON TIME
BELT DIR - FRW
BELT DIR - REV
00.000 Sec
F1
F2
F3
F4
3_Setting_Stepper_Motors [Screen]


Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 4_IO_LIST_DI
SIEMENS
SIMATIC HMI
12/31/2000
IO LIST
10:59:39 AM
TOUCH
CPU - DIGITAL INPUT
0.0 MANUAL AUTO SWITCH
0.1 STRIP DETACTION PROXY
DO >>
F1
F2
F3
F4
4_IO_LIST_DI [Screen]

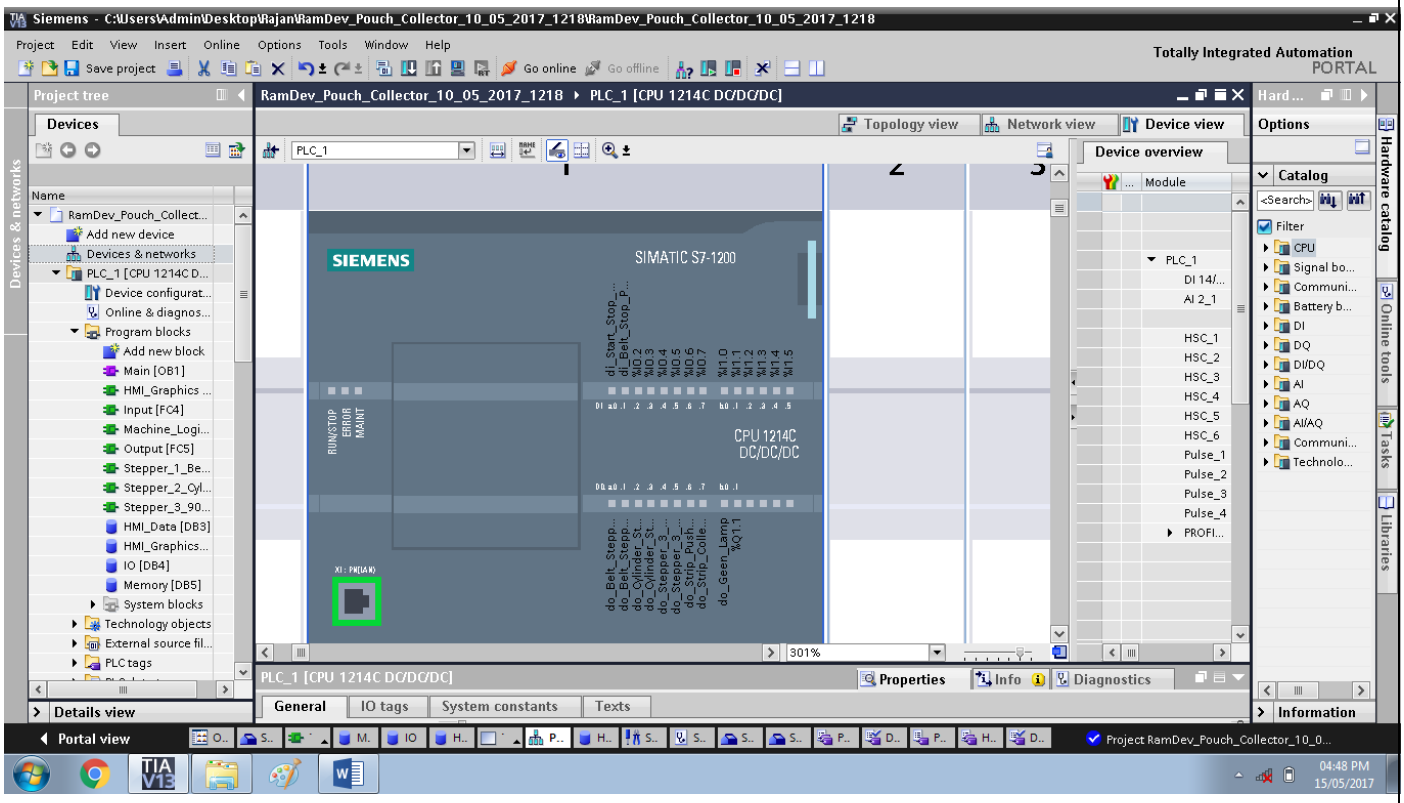

Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 4_IO_LIST_DO
SIEMENS
SIMATIC HMI
12/31/2000
IO LIST
10:59:39 AM
CPU - DIGITAL OUTPUT
0.0 BELT MOTOR PULSE
0.1 BELT MOTOR DIRECTON
0.2 CYLINDER MOTOR PULSE
0.3 CYLINDER MOTOR DIRECTON
0.4 STEPPER MOTOR 3 PULSE
0.5 STEPPER MOTOR 3 DIRECTON
0.6 PUSH VALVE
0.7 CYLINDER VALVE
<< DI
TOUCH
F1
F2
F3
F4
4_IO_LIST_DO [Screen]
Properties
Info
Diagnostics
Portal view
Project RamDev_Pouch_Collector_10_0...
04:43 PM
15/05/2017


Siemens - C:\Users\Admin\Desktop\Rajan\RamDev_Pouch_Collector_10_05_2017_1218\RamDev_Pouch_Collector_10_05_2017_1218
Totally Integrated Automation PORTAL
RamDev_Pouch_Collector_10_05_2017_1218 ▸ HMI_1 [KTP400 Basic PN] ▸ Screens ▸ 5_Prog_Step
SIEMENS
SIMATIC HMI
12/31/2000
10:59:39 AM
No.
Date
Time
Text
TOUCH
F1
F2
F3
F4
5_Prog_Step [Screen]
Properties
Info
Diagnostics
Portal view
Project RamDev_Pouch_Collector_10_0...
04:44 PM
15/05/2017

17). PlC-HMI Connection



18). PLC Connections



19). HMI Connection