

NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

REMOTE PROCESS CONTROL

Graduation Project EE - 400

Student: Ömer Akkanat (2001366)

Supervisor: Özgür Cemal Özerdem

Lefkoşa - 2002

TABLE OF CONTENTS

ACKNOWLEDGEMENT	1
	П
ABSTRACT	**
INTRODUCTION	ш
MAIRODOCATOR	
1. INTRODUCING THE PLC	1
1.1. What is PLC?	1
I. The Central Processing Unit	1
II. The Input/Output System	1
1.2. Inputs And Outputs	2
1.3. Control Programs	3
1.4. How PLC Work?	4
1.5. Why Use PLC's?	6
1.6. Capabilities of the S7-212 CPUs	6
1.7. SIMATIC S7-200 Quick Reference Card	13
2. PROCESS	15
2.1. Introduction	17
2.2. Parallel Port Controlling program	17
2.3. Electronic Circuit	17
2.4. Server Environment	10
I. Direct	10
II. Modem	10
III. Network	10
IV. Internet	18
2.5. Communication and Client Environment	19
2.6. Connection for Chat	20
2.7 Process and Alarm	20

	LIBRAD
3. PRODUCTION UNIT	al too
3.1. Introduction to the Production Unit	21 J. LEFKOSP
3.2. Air ventilation fan	22
3.3. Silo 1.	22
3.4. Silo 2.	22
3.5. Nylon bag production unit	23
3.6. Sensors	23
4. PORT CONTROL UNIT4.1. Parallel Port Control	24
4.2. Port Register	24
4.3. Parallel Port Control Circuit (PPCC)	25
4.4. Parallel Port Control Program	27
5. PLC PROGRAMS	
5.1. Introduction	61
5.2. Ladder Diagram	63
5.3. Statement List	77
CONCLUSION	84
CONTOLIONAVA	
DEFERENCES	85
2	

ACKNOWLEDGEMENT

First I want to thank Mr. Özgür Cemal Özerdem to be my Supervisor. Under his guidance, I successfully overcome many difficulties and learn a lot about PLC with EE 470 PLC course. PLC depends greatly on technology. In each discussion, he explained my questions patiently and answered my questions quickly and in detail. He always help me a lot either in my study.

Thanks to faculty of Engineering for having such a good computational environment. I want to thank all my student friends in NEU. Being with them make my during educational in NEU full of fun.

I also special thanks to Mr. Aziz Zenginses with their kind help. I could successfully to perform computational problems. He always help me a lot either in my studys

Finally I want to thank especially my parents. Without their endeless support and love for me. I wish daughter in the heaven be proud of me.

ABSTRACT

In this study, a system of Remote Process Control (REPCO) via telephone line with modem, network or internet is introduced. Minimum requirements to use this system are two computers: a server and a client connected through modem.

In this environment, the role of the server is to control the process via some sort of controller device. On the other hand, the client is to convey the control related commands and messages of the user to the server. Therefore, as long as there is a modem or network connection, the client can be anywhere.

In our research, we have used a Programmable Logic Controller (PLC) to transmit the equipment's data to the server computer and control the equipments by executing the commands received. In our approach, the server will process the incoming equipment related data and will send them, along with the associated screen snapshots, to the client. Server will also transmit the user and/or client-originated commands to the PLC. The fundamental idea of getting data and screen snapshots from the server and responding by sending the required commands will allow user/client to obtain the control authorization for monitoring the process remotely. Although the scope of this experiment contains a small infrastructure composed of one process, one server, and one client, the proposed method can easily be adapted to systems in different domains.

INTRODUCTION

The rapid developments in electronics have unstoppably revolutionized the Internet technology in recent years and have become a part of daily life, bringing in everything from Pampers to programmable logic devices. The rising usage of Internet has been complemented with developments in communication technology.



Fig. 1 Siemens S7 CPU212 PLC.

Moreover, Networking technologies start to invade the home to carry phone signals and TV programs, link computers and peripherals, and tap into the Internet (Data-Ray, 1999a and Dutta-Ray, 1999b). These rapid developments show that Internet based systems have had a huge role in home and business solutions nowadays and will have more in future. Therefore, many software developers have been forced to find ways of developing systems more quickly than ever before.

Internet technology allows companies to overcome many of the physical constraints that often prevent them from doing business in distant markets, which means that an a commerce market is fundamentally global (Choi and Whinston, 1999).Communication Systems for Next-Generation Remote Education Using Asymmetrical Satellite and Terrestrial Networks (Yoshida et al., 1999) are a few examples of applications on Internet. Besides, in a recent study, a number of researchers (Mccalley et al., 1998) explained that Internet technology provides a way to power engineering educators globally to combine their efforts in increasing course development resources.

For years research has focused on ways to allow remote access via standard communication. With the growth of Internet, one finds more and more devices such as coffee machines, telescopes, manipulators, and mobile robots connected to it. Open Access to a Mobile Robot on the Internet and Mobile Robots in Public Places are examples of remote control (Saucy and Mondana, 2000). A Feasibility Study for Internet Robots shows that controlling over Internet is becoming widespread.

The issue of process control has been effectively instrumental in raising the productivity in industrial automation for years. Remote control of processes is new era and only supported by a few companies, which are the giants of automation industry. Our proposal is also related to the control of industrial process with the exception of Internet. The idea proposed here comes with industrial automation for accessibility and controllability from anywhere.

The goal of this study is to discuss the details of the system that can control an industrial process over the Internet or modems with telephone lines. The remote processing has a bright future, especially in the companies with many branches where electronically controlled machines are deployed as a part of the workforce. While a computer connected to the automation devices can control and take data from processes at a production unit, the operator can process the data according to the rules and regulations and then issues commands necessary to control the system.

	Communicati	on	Number of Process Connection	Modification
Commercial	Server	Client	Server	Client
Methods	Communicates with the client using web servers such as PWS TM , IIS TM	Using browsers, add-ons and software	Can connect only a process except Factory Suite	When an addition or changing is needed in process, they have to update/change server and client software
Proposed Method	No need to use web server	Only setup client software is enough	A client can connect more than one process	Changing server software is enough. When an error is occurred in server side, it could be corrected using mouse, keyboard, and screen utility at the client user.

Table 1 Characteristic differences in remote processing.

Technical Approach: For simulation purposes, Siemens S7 PLC process controller, shown in Figure 1, is used to control the processes of the equipments, and to bi-directionally communicate with the server. Additional tools are also developed to maintain the process control dialog between the server and the client.

Although some features of our method are similar to the methods available commercially in the industry, connecting to more than one process without using a web server is the unique feature of our approach. In other words, as far as monitoring and local control are concerned, the architecture of our work can be compared with the following software tools: WinCC from Siemens (Simens). For the remote control, the comparable software is PcANYWHERE from Symantec (Symatec).

Also our proposed monitoring and local control section will display a behavior similar to those mentioned above, the substantial difference will be in the remote control section as shown in Table 1. Therefore, we believe that the proposed work is an economic and useful solution for large and distributed companies, and workshops, which have to work together, even for institutions having health-critical or hazardous jobs.

The Architecture: A computer connected to the automation devices can control and get data from processes at a production unit. In the REPCO environment introduced this process is done using PLC connection to a server, which can be controlled via telephone line or internet by a user who is far away from unit, as shown in Figure 2.

The REPCO environment can be technically decomposed into three parts:

- I. Process
- II. Server Program
- III. Client Program.

The high level functional flow of the system is shown in Figure 3. The server software part is designed to handle three jobs. First it establishes a communication path with PLC. Second, the server maintains a smooth operation and records process related data. Lastly, it communicates with the client to obtain the process data and the server screen, according to which user commands are fed to the server. The communication between server and Client is handled through the use of Windows Socket, which supports TCP/IP. Note: Since HTTP and HTTPS1 protocols support only the connectionless communication, HTTP and related protocols are not considered at this stage.



Fig. 2 User process connections.

1. INTRODUCING THE PLC

1.1 What is PLC?

A PLC (programmable logic controller) is a small industrial computer which originally replaced the necessary sequential relay circuits for machine control. The PLC works by looking at its inputs and depending upon their state, turning on/off its outputs. It contained a program which executed a loop, scanning the inputs and taking actions based on these inputs. The user enters a program, usually via software, that gives the desired results.

PLCs are used in many "real world" applications. If there is industry present, chances are good that there is a PLC present. If you are involved in machining, packaging, material handling, automated assembly or countless other industries you are probably already using them. If you are not, you are wasting money and time. Almost any application that needs some type of electrical control has a need for a PLC.

A PLC, basically consists of two elements:

- I. the central processing unit
- II. the input/output system

I. The Central Processing Unit

The central processing unit (CPU) is the part of a programmable controller that retrieves, decodes, stores, and processes information. It also executes the control program stored in the PLC's memory. In essence, the CPU is the "brains" of a programmable controller. It functions much the same way the CPU of a regular computer does, except that it uses special instructions and coding to perform its functions.

The CPU has three parts:

- I. the processor
- II. the memory system
- III. the power supply

The processor is the section of the CPU that codes, decodes, and computes data. The memory system is the section of the CPU that stores both the control program and data from the equipment connected to the PLC. The power supply is the section that provides the PLC with the voltage and current it needs to operate.

II. The Input/Output System

The input/output (I/O) system is the section of a PLC to which all of the field devices are connected. If the CPU can be thought of as the brains of a PLC, then the I/O system can be thought of as the arms and legs. The I/O system is what actually physically carries out the control commands from the program stored in the PLC's memory.

The I/O system consists of two main parts:

- I. the rack
- II. I/O modules

The rack is an enclosure with slots in it that is connected to the CPU. I/O modules are devices with connection terminals to which the field devices are wired. Together, the rack and the I/O modules form the interface between the field devices and the PLC. When set up properly, Each I/O module is both securely wired to its corresponding field devices and securely installed in a slot in the rack. This creates the physical connection between the field equipment and the PLC. In some small PLC's, the rack and the I/O modules come prepackaged as one unit

1.2. Inputs And Outputs

All of the field devices connected to a PLC can be classified in one of two categories:

- I. inputs
 - II. outputs

Inputs are devices that supply a signal/data to a PLC. Typical examples of inputs are push buttons, switches, and measurement de-vices. Basically an input device tells the PLC,

"Hey, something's happening out here...you need to check this out to see how it affects the control program.

Outputs are devices that await a signal/data from the PLC to perform their control functions. Lights, horns, motors, and valves are all good examples of output devices. These devices stay put, minding their own business, until the PLC says, "You need to turn on now" or "You'd better open up your valve a little more," etc.

There are two basic types of input and output devices:

- I. discrete
- II. analog

Discrete devices are inputs and outputs that have only two states: on and off. As a result, they send/receive simple signals to from a PLC. These signals consist of only 1's and 0's. A 1 means that the device is on and a 0 means that the device is off.

Analog devices are inputs and outputs that can have an infinite number of states. These devices can not only be on and off, but they can also be barely on, almost totally on, not quite off, etc. These devices send receive complex signals to from a PLC. Their communications consist of a variety of signals, not just 1's and 0's. Because different input and output devices send different kinds of signals, they sometimes have a hard time communicating with the PLC. While PLC's are powerful devices, they can't always speak the "language" of every device connected to them. That's where the I/O modules we talked about earlier come in. The modules act as "translators" between the field devices and the PLC. They ensure that the PLC and the field devices all get the information they need in a language that they can understand.

1.3. Control Programs

We talked a little bit earlier about the control program. The control program is a software program in the PLC's memory. It's what puts the control in a programmable controller. The user or the system designer is usually the one who develops the control program. The control program is made up of things called instructions. Instructions are, in essence, little computer codes that make the inputs and outputs do what you want in order to get the result you need. There are all different kinds of instructions and they can make a PLC do just about anything (add and subtract data, time and count events, compare information, etc.). All you have to do is program the instructions in the proper order and make sure that they are telling the right devices what to do and voila!...you have a PLC-controlled system. And remember, changing the system is a snap. If you want the system to act differently, just change the instructions in the control program. Different PLC's offer different kinds of instructions. That's part of what makes each type of PLC unique.

However, all PLCs use two basic types of instructions:

- I. contacts
- II. coils

Contacts are instructions that refer to the input conditions to the control program that is, to the information supplied by the input field devices. Each contact in the control program monitors a certain field device. The contact waits for the input to do something in particular (e.g., turn on, turn off, etc. this all depends on what type of contact it is). Then, the contact tells the PLC's control program, "The input device just did what it's supposed to do. You'd better check to see if this is supposed to affect any of the output devices."

Coils are instructions that refer to the outputs of the control program that is, to what each particular output device is supposed to do in the system. Like a contact, each coil also monitors a certain field device. However, unlike a contact, which monitors the field device and then tells the PLC what to do, a coil monitors the PLC control program and then tells the field device what to do. It tells the output device, "Hey, the PLC just told me that the switch turned on. That means that you're supposed to turn on now. So let's go!" In PLC talk, this three-step process of monitoring the inputs, executing the PLC control program, and changing the status of the

1.4. How PLC Work?

A PLC works by continually scanning a program. We can think of this scan cycle as consisting of 3 important steps. There are typically more than 3 but we can focus on the important parts and not worry about the others. Typically the others are checking the system and updating the current internal counter and timer values.



Step 1-CHECK INPUT STATUS-First the PLC takes a look at each input to determine if it is on or off. In other words, is the sensor connected to the first input on? How about the second input? How about the third... It records this data into its memory to be used during the next step.

Step 2-EXECUTE PROGRAM-Next the PLC executes your program one instruction at a time. Maybe your program said that if the first input was on then it should turn on the first output. Since it already knows which inputs are on/off from the previous step it will be able to decide whether the first output should be turned on based on the state of the first input. It will store the execution results for use later during the next step.

Step 3-UPDATE OUTPUT STATUS-Finally the PLC updates the status of the outputs. It updates the outputs based on which inputs were on during the first step and the results of executing your program during the second step.

1.5. WHY USE PLCS?

The software advantage provided by programmable controllers is tremendous. In fact, it is one of the most important features of PLCs. Software makes changes in the control system easy and cheap. If you want a device in a PLC system to behave differently or to control a different process element, all you have to do is change the control program. In a traditional system, making this type of change would involve physically changing the wiring between the devices, a costly and time-consuming endeavor. In addition to the programming flexibility we just mentioned, PLCs offer other advantages over traditional control systems. These advantages include:

- high reliability
- small space requirements
- computing capabilities
- reduced costs
- ability to withstand harsh environments
 - expandability

1.6. Capabilities of the S7-212 CPUs

The S7-200 family includes a wide variety of CPUs. This variety provides a range of features to aid in designing a cost-effective automation solution. Table 2 provides a summary of the major features of each S7-200 CPU.

ALC: NOT STREET

Table 2

CPU 212 DC power supply, DC inputs, DC outputs

Order Number: 6ES7 212-1AA01-0XB0

General Features		Output Points (continued)		
160 x 90 x 62 mm	Switching dalay	25 ps ON, 120 µs OFF		
(6.3 x 3.15 x 2.44 in.)	Surge current	4 A, 100 ms		
0.3 kg (0.7 lbs.)	Voltage drop	1.8 V maximum at maximum		
5 W at 1.75 A load		current		
512 words/EEPROM	Optical isolation	500 VAC, 1 min		
512 words/RAM	Short circuit protection	None		
50 hr typical (8 hr ministraum at 40° C)	Input Points			
\$ inputs% outputs	Input type (IEC 1131-2)	Type 1 sinking		
2	ON state range	15-30 VDC, 4 mA minimum 35 VDC, 500 rus surge		
64 juputs/64 outputs	ON state norminal	24 VDC, 7 mA		
16 juputs/] 6 outputs	OFF state maximum	5 VDC, 1 mA		
1.2 pastastruction	Response time	0.3 ms maximum		
128	restical isolation	500 VAC. 1 min		
64 timers	Power Sunnly			
64 counters	Yahas ruse	20 4 to 28 8 VDC		
1 software (2 KHz max.)	s onage tringe	60 mA innical CPU only		
1	mpan content	500 mA maximum load		
UL 508 CSA C22.2 142	UL/CSA rating	50 VA		
FM Class I, Division 2 VDE 0160 compliant CE compliant	Heldup time	10 ms minimum from 24 VDC		
	hrush current	10 A peak at 28.8 VDC		
Some ing transistor	Fusing (non-replaceable)	LA, 125 V. slow blow		
26.4 VDC to 28.8 VDC	5 VDC current	260 mA for CPU 340 mA for expansion I/O		
0.1040° C 55°C ² 0.75 A 0.50 A	Isolated	No		
1.00 A 0.75 A	DC Sensor Supply			
avant fin ter a me	Voltage range	16.4 to 28.8 VDC		
2A L/R = 10 ms	Ripple noise (<10MHz)	Same as supplied voltage		
LA L/R = 100 ms 1 W energy dissipation	24 VDC available current Short-circuit current limit	180 mA < 600 mA		
(1/21.P X SWIGHTER < 1 W)	I STORE STORE STORE STORE			
	160 x 90 x 62 mm (6.3 x 3.15 x 2.44 in.) 0.3 kg (0.7 lbs.) 5 W at 1.75 A load 512 words/EEPROM 512 words/EAM 50 hr typical (% hr minimum at 40° C) \$ inputs/6 outputs 2 64 inputs/64 outputs 16 inputs/64 outputs 12 geinstruction 128 64 timers 64 counters 1 woliwate (2 KHz max.) 1 UE 508 CSA C22.2 142 FM Class I, Division 2 VDE 0160 compliant CE compliant CE compliant CE compliant Souncing transistor 20.4 VDC to 28.8 VDC 04.0 A 0.75 A 2.25 A 1.75 A (per common) 2A L/R = 10 ms 1A L/R = 100 ms 1/2 L/P x switch rate < 1 W)	160 x 90 x 62 mm (6.3 x 3.15 x 2.44 in.)Switching dzkay0.3 kg (0.71bs)Switching dzkay5 W at 1.75 A loadSorge current512 words/REPROMOptical isolation512 words/RAM 50 hr typical (8 hr minimum at 40° C)Optical isolation5 inputs/6 outputsInput Points2ON state maximum64 inputs/6 outputsOFF state maximum16 inputs/6 outputsOFF state maximum12 usinstructionIOF state maximum12 usinstructionIOF state maximum12 usinstructionPower Supply64 countersVeltage range1 wit Sos CSA C22.2 142Weltage range1 wit Sos CSA C22.2 142Weltage range1 wit Someing transistorULCSA rating20.4 VDC to 28.8 VDCSVDC current0 to 407 C55%C20.75 A0.50 A1.00 A0.75 A2.25 A1.75 A(pr commoti)Cature of the signation1 We nergy dissipationUC svaitable current1 Sourcing transistorSVDC current20.4 VDC to 28.8 VDCSVDC current0 to 407 C55%C20.75 A0.50 A1.00 A0.75 A2.25 A1.75 A0 to apply visipationVoltage range1 W energy dissipationRipple/noise (<10MHz)		

1 The CPU reserves 8 process-image input and 8 process-image output image register points for local I/O.

Linear derate 40 to 55° C. Vertical mount derate 10° C.



Connector Terminal Identification for CPU 212 DC/DC/DC

CPU 212 AC Power Supply, DC Inputs, Relay Outputs

er Number: 6ES7 212-1BA01-0XB0

eneral Features		Input Points		
"leysical size (L x W x D)	160 x 80 x 62 mm	Input type (IEC 1131-2)	Type 1 sinking	
	(6.3 X 3.1 5 X 2.49 III.)	ON state range	15-30 VDC, 4 in A minimum 35 VDC, 500 mis surge	
Weight	0.4 Kg (0.9 IBK)	(N) state marring)	24 VIX: 7 mA	
Power dissipation	6W	CAN SALE DOUBLET	SADC LOA	
i ser program size/storage	512 words/EEPROM	OFF State Haasarium	2 110.5 1 10.5	
User dala size/storage Data refention	512 words/RAM 50 hr typical (8 hr minimum at 40° CS	Response time 10.0 to 10.7	0.3 ms maximum	
Local I/OI	S inputs/6 outputs	Optical isolation	290 971-, 1 800	
Maximum number of	2	Voltage/frequency range	85 to 264 VAC at 47 to 63 Hz	
expansion modules Digital I/O supported	64 inputs/64 outputs	Input current	4 VA typical, CPU only 50 VA maximum load	
Analog 1/O supported	16 inputs 16 outputs	Haddens time	20 ps minimum from	
Boolean execution areed	1.2 perinstruction	madup mue	HUVAC	
Internal momory bits	128	Inrush current	20 A peak at 264 VAC	
Timets	64 timers	Fusing (non-replaceable)	2 A. 250 V. slow blow	
Counters	64 counters	5 VDC current	260 mA for CPU 340 mA for expansion EO	
High-speed counters	1 software (2 Kllz max.)	Isolated	Yes, Transformer, 1500 VAC	
Analog adjustments	1	Provide Ci	l min	
Standards compliance	UL 508 CSA C22.2 142	2 142 DC Sensor Supply		
	VDE 0160 compliant CE compliant	Voltage range	20.4 to 28.8 VDC	
Output Paints		Ripple noise (< Huwhiz)	180 mA	
Output type	Relay, dry contact	Shori circuit current limit.	< 600 mA	
Voltage range	5 to 30 VDC/250 VAC	Isolated	No	
Maximum load current	2 A/point, 6 A/commen			
Overcurrent surge	7 A with contacts closed			
Isolation resistance	100 MQ minimum (new)			
Switching delay	10 mis maximum			
Lifetime	10,000,000 mechanical 100,000 with rated load		-	
Contaci resistance	200 mΩ maximum (new)			
Isolation coil to contact contact to contact (between coen contacts)	1500 VAC, 1 min 750 VAC, 1 min			
Short circuit protection	None			

1 The CPU reserves 8 process-image input and 8 process-image output image register points for local 1/0.



Connector Terminal Identification for CPU 212 AC/DC/Relay

ieneral Features		input roinis	
hysical size (1. x W x D)	160 x 90 x 62 mm	Inpui iype (IEC 1131-2)	Type 1 sinking
Uniolat	(6.5 X 3.1 5 X 2.44 III.) 0.4 kp (0.9 lbs.)	ON state range	15-36 VDC, 4 mA minimum 35 VDC, 500 ms surge
nulyin	6W	ON state norminal	24 VDC, 7 mA
ower cussipation	All and All DOCA	OFF state maximum	5 VDC, 1 mA
lser program size/skrage	212 WORKSPEP RUMA	Doctores time	C. In Concession
User data size/storage Data retention	512 words/RAM 50 hr typical (8 hr minimum at 40° C)	10.0 to 19.7	0.3 ms maximum
Contract of the second s	Constitution and so the	Optical isolation	500 VAC, I shat
Local POP	o mburza cambaro	Power Supply	
Maximum number of expansion modules	2	Voltage/frequency tange	20 to 29 VAC at 47 to 63 Hz
Digital 10 supported	64 inputs/64 outputs	Input current	4 VA typical, CPU only 50 VA maximum load
Analog I/O supported	16 inputs/16 outputs	Heldup time	20 ms minimum from 24 VAC
Boolean execution speed	1.2 µs instruction		20 A mode of 20 MA
Internal memory bits	128	Inrush current	20 A peak at 29 VAL
Timers	64 timers	Fusing (non-replaceable)	2 A. 250 V. slow blow
Comlets	64 counters	5 VDC current	260 mA for CPU 340 mA for expansion DO
High-speed counters	l software (2 Kllz max.)	Isolated	Yes. Transformer, 500 VAC,
Analog adjustments	1		min
Standards compliance	UL 508 CSA C22.2 142 FM Class 1 Division 2	DC Sensor Supply	
	VDE 0160 compliant CE compliant	Voltage range	26.4 to 28.8 VDC
Output Points		Ripple noise (< mining)	180 mA
Output type	Relay, dry contact	Short circuit current limit	< 600 mA
Voltage range	5 to 30 VDC/250 VAC	Isolated	No
Maximum load current	2 A /point, 6 A /common		
Overcurrent surge	7 A with contacts closed		
Isolation resistance	100 MO minimum (new)		
Switching delay	10 cus maximum		
Lifetime	10,000,000 mechanical 190,000 with rated load		
Contact resistance	200 mQ muximum (new)		
Isolation coil to contact contact to contact (between open contacts)	1500 VAC, 1 min 750 VAC, 1 min		

24 VAC CPU 212 Power Supply, DC Inputs, Relay Outputs

1 The CPU reserves 8 process-image input and 8 process-image output image register points for local 1/0.

None

Short circuit protection

] [



Connector Terminal Identification for CPU 212 24 VAC/DC/Relay

1.7. SIMATIC S7-200 Quick Reference Card

Priority Group	Exent	Description		Priority in Green
	8	Port 0: Receive character		0
	9	Port0: Fransmit	回回到周	0
Contraction references	23	Porto: Receive melisage	国国	0
Historie priorieg	24	Port 1: Receive mensage	E.	1
	26	Port 1: Receive character	30	1
	26	Port 1: Transmit complete	.St.	1
	0	Rising edge, 10.0*	and and	0
	2	Rising edge: 10.1	DEL	1
	4	Rising edge, 19.2	E III A	2
	6	Rising edge, 10.2	112131	3
	1	Faling edge, \$0.0"		4
-	3	Faling edge, KA1	and an	5
	5	Faling edge, 10.2	四川側	0
1. Lintern sets	7	Faling edge, 10.3	27.01.41	7
Idde windty	12	HS00 = preset value*	四四四四	Ö
and burney	15	HSC1 = preset value	তারার	8
	14	RECT desction change	리고에	9
	15	HSG1 external read	13 23 31	10
	16	HSC2 = preset value	四四月	11
	17	HSC2 desction change	10131	12
	18	HSC2 esternal reset	ana	18-
1	19	PLEM	記録の	14
	20	PLS1	2233	15
	10	Timed0	10.02.03.03	0
Timed interrupts	11	Timed 1	12 14 51	1
several primity	21	T32 = preset	13.33	2
	100	1245	The last	

	Speci	ol Mamory E	Bits
9.6612	Always On	SM1.0	Result of operation = 0
S 10.1	First Scan	8141.1	Question or illegal value
5140 /	Retentive data los	SI412	Negative result
EW9.1	Power up	SI 11 3	Division by 0
\$140.4	30 s oli / 30 s on	8141.4	Table full
SI 10 6	0.5 s off /0.5 s on	SI(1) 5	Table compty
9 10.6	Gil 1 scan/os 1 scan	SM1.6	BCD to broary conversion error
SMD.7	Switch in RUN position	EMIT.	ASC # to hex conversion outor

	High-Speed	Counter	Viocios		
	Counter	T	Inp	uls	
HSCO	blasimum 2 kHz (1) 2121 -1	IQ.D			
HSC1	7 HHz III 20 KHz III	10.6	10.7	11.0	11.1
HBC	7 13-12 [1] 20 kHz [1]	11.2	11.3	31.4	11.5
Mode	Description	C	ock	Roset	Start
0 10 2	Single phase with internal direction	Up/Down: 9, 1, 2	1	1,2	2
3 to 5	Sugle phase with external direction	Up/Down: 3, 4, 5	Din tion 3, 4, 5	4,5	Б
i ka B	Teophne	8p: 6.7.8	Down: 0, 7, 6	7,8	8
9 10 11	Quadrature AlB	A: 9. 10, 11	8: 9, 10, 11	10.11	11
I CPU	212 CPU, 214 CI	PU 215 31	CPU 215		A

Description		Ra	nge Limit			Accessi	blo as	
seconpriori	212	214	215	216	Bit	Byte	Word	DWord
Uner Program Size	512 W	2045 W	40% W	1000 W				
Uner Dala Size	512 W	2048 W	2500 W	2500 W	-			
Voriable menory	A 1623	0-4095	0.5119	0.5119	Mary.	Whe	VSAJ.	MD
Input Image Register	0.7	0.7	6.7	0.7	lr.e	ally a	EAL	V L2k
Output lineare Register	0.7	0.7	0.7	0.7	Chev	OBr	OW	004
Analog liiputs	0.30	00	0.30	0.10			ABA	
Analog Outputs	0.30	0.30	0.30	0.30	-		ACM2x	
Bit Memory	0.15	0.31	0.31	0.31	Max	LID.	LEUAL	LD.
Special Memory	0.45	6.85	0.194	0.194	SAA 1	SLED	53.5.N.	SND
Referrive liners 1 ans	0	0,64	0.64	01	Tr	Cristian	Ty	-9611.72
Referitive Timers 10 ms	1.4	14.65.0	1-4, 05-68	14.65.68	Te	-	Te	
Paterlive limers 100 m	6.31	4.31, 02.95	5.31, 69.95	531, 69.95	Tr		14	
On Delay Timers 1 ms	32	32.96	32.96	32.96	Tr		Te	
On Doby Tearrant 10 mis	33.36	33-36, 97-100	31.30, 97-100	33.98.97.100	Tr.	-	T.	
On Dalay Timers 100 ms	37-63	37-03, 101-127	37-03, 101 256	17.63.101.255	Tr		Tu	-
Counters	0.0	0.127	0.265	0.265	0		1.5	- 1
High Speed Counter	0	0.2	6.2	0.9				LIPS-
Accusedators	0.3	£0	0.3	6.1		10	A2-	HL.a
Sequence Control Reby (SCR)	0.7	0.15	11.0	0.01	Su u	EN.	200	14.8
Jumps Labels	0.63	0.255	0.255	1.255	we'y	COL	CHIFT	34.74
Cal Sulacutine	0.15	0.63	0.03	0.63				
Interrupt Routines	0.31	0.127	0-127	0.127	4			
frame upt Evenue	0,1.8-10, 12	0.20	0.25	0.10	1			
PID Loops	NA	NA	6.7	10.7				
Parts	Par 0	Pon #	Fut 0. DP Part	Port 6 Port 1				

	Book	ant	nstructions
-	H		Lond
M	н		Load travedinte
104	н	1	Load Not
000	N		Lond Not Innevedial
*	H		AND
×	H		AND Immediate
100	N		AND Not
-	н		AND Not know disk
2	N		DIR
1K -	-		CR Immediate
144	11	- 1	CR Not
102	н		OR Not immediate
Line	N1. H2		
LTB-	H1, N.F		Lose would of Byte Company
-	111, 108		Bit (+, ++, or set inc
10	H1.N2		
ALC:	N1. N2		AND result of Byte Compare
100	111.10		141 (E, 22, 01 SE) 162
Contra Contra	111 11		
CHER -	111.112		OR result of Dyle Compare
CHIEF.	H1 H2		NI (=, >=, or <=) H2
I Party	111.122		
	111, 044		Lond roout of Word Company
LOW	111, 102		Nt (=, ≥=, or <=) H2
LUID	101,104		
Real Provide State	H3, H2		AND mesult of Word Compare
ALC: N	H1, H2		N1 (=, >=, or <=) 112
Art	111, 10/		
041=	H1, H2		CR much of Word Company
1141-3	H1. H2	1	N1 (=, >=, or <=) H2
-04/	H1, H2		
ADD	H1, H2		Loud start of Diffiel Company
LD0 =	H4, N2		N1 i=, >=, or <= i H2
100 -	N1. NZ		
AD-	H1, H2		
ADPE	N1. N2		AblD result of DWord Company N1 (c. 2c. or Set 112
AD<=	H1, H2		
OD=	11, 112		
00=	H1. N2		NI (=. >=. or <=) H2
OD::	N1. N2		ter full that they
LDR	N1. N2	22	
LOR =	811. N#		Lond result of Roal Compare
LUS	H1. N2	-	DH IS, WA, OF SALIDA
AR=	111.112	08	
AR>=	111. NR	11	AND result of Read Compare
612 -	H1 NV	100	NI (=, >=, or <=) NZ
DD	114 182		
OlDun	114 115	744	OR result of Real Compare.
DR-	111 N2		Nt (=, >=, or <=) N?
6400	111, 114	-	Read Blonation
TAL I			Datastica of Dising Edu
10			Detection of Falling Edge
10			Descarar or noning trailer
=	N		Ameria Vakio
H I	N		Ansight Value Introducto
8	S DII, N		set bit Range
R	SDIEN		Reset bit Range
3	S BIT, N		Set bit Range Insuresting
RI	S BIT, N		React bit Ran jo transdiate
	Math. Incr	eme	nt. and Decrement
		Inst	ructions
01	INI, OUT		Add Internet (Manual or Rend
+D	IN1, OUT		INTACALL CALL
+R	INI, OUT	d.	
1	INI, OUT		Subtra University Official on
-40	INI, OUT		Roal
-16	INI, OUT	13	COLINISCUT
7.8.8	INI. OUT	-	Multiply late per or Real
R	INI. OUT	415	NI 'OUT = CUT
135/	INS CHILF		Divide Internet Pi-1
42	INI CHIT	-	Divide interfer of Head
114	141,001	-	

NUL	IN, 170 I	121 4	State of the second sec
NUL		~	Scinne Level
DAL LAD	001	m	In some of Ret- Wood or
R. 10. 74.	OUT		DWord
NCD	OUT		
DICB	our	21	
DECW	OUT	1	Discrement Byte Word, or
DECD	OUT		P.resta
PD	Table Loop	101	PIDLoop
	1000000000	1100	inter lo truttions
100.000	Timer and	CN	Die San
TON	Door PT		On Delay Timer
TOUR	1000, PT		Relentric Unitieslay times
CTU	Cox PV		Court Up
GIUD	Cox PV	-	Gour UpDown
	Real Tim	E CI	ock instructions
TODE	T	-10	Read Time of Day clock
TODW	T	710	White Tisse of Day cluck
TODAY	Ore creates	Can	Prolingtrugtions
	Program	UGE	cror moundations
END			Ganditional First of Program
INER			Main Program End of Program
STOP			Transition to STOP Mode
VIDR			WetchiDog Rosel (208 ave)
HAP	N	-	Jump to defined Labor
1.69	N		Define of the Jonn to
CMI	N	_	Call a Subsection
CALL	N N		Define a Coloradia ta ba
STAR	N		Called
CHET			Conditional Faltern from SBR
RET			Unconditional Linears from
			EIR
Frank.	index Initial.		
	Final		For Next Loop
NEXT			
LSCR	N		band Emerities and East
SURT	N		Sequence Grainal Relay
SCIL			Segueri
- Plan	CINH D	at at	and Sill Instructions
1050-1	C. Still, P	oun	, and a more denoted
FROME	IN, OUT		
ALC: NO REAL			
	14, 100		Maxie Byte, Word, D'Alord, Real
NOVO	IN, OUT		Mave Byle, Word, DWord, Real
MOVE	IN, OUT	101	Marie Byle, Word, D'Alord, Real
MCVD MCVR EMB	IN, OUT IN, OUT IN, OUT	14	Move Byb. Word, D'Word, Real
MCVD MCVR EMB EMW	IN, OUT IN, OUT IN, OUT IN, OUT, H	14	Mare Byte, Word, D'Alord, Read
MCVD MCVR EMB EMV EMD	IN, OUT IN, OUT IN, OUT IN IN, OUT, IN IN, OUT, IN	21	Move Byte, Word, D'Aord, Real Elizah Move Pyte, Word D'Word
MCVD MCVR EMB EMW EMD	IN, OUT IN, OUT IN, OUT II IN, OUT, II IN, OUT, II IN, OUT, II		Move Byte, Word, D/Vard, Read Elical: Move Dyte, Word, DWard
MCVD MCVR EMB EMV EMD SWVP	IN, OUT IN, OUT IN, OUT IN IN, OUT, IN IN, OUT, IN IN, OUT, IN IN		Move Byte, Word, D'Alard, Beat Electric Move Fighte, Word, D'Word Swap Dytes
MCVD MCVR EMB EMW EMD SWWP SHID	N. OUT IN. OUT IN. OUT IN. OUT. H IN. OUT. H IN. OUT. H IN.	21 ,N	Move Byb. Word, D'Aard, Read Block Move Fyte, Word, DWord Swap Dyte: Shall Register Bit
MOVD MOVR EMB EMB EMV EMD SWAP SHI31 SRD	IN, OUT IN, OUT IN, OUT, H IN, OUT, H IN, OUT, H IN Data, S, ba OUT, N	21 21 ,N	Move Byte, Word, D'Mard, Read Block Move Byte, Word, D'Word Swap Dyte: Shait Register Bit
MCVD MCVR EMB EMB EMV EMB SMWP SHR31 SRM SRM	IN, OUT IN, OUT IN, OUT, H IN, OUT, H IN, OUT, H IN Data, S, 54 OUT, N OUT, N	ы 23 м,	Move Byte, Word, D'Alard, Read Elastic Move Fighte, Word, D'Alard Shall Register Bit Shall Register Bit
MCVD MCVR EMB EMB EMB EMB EMB SMM SRM SRM SRM SRM SRM	IN, OUT IN, OUT IN, OUT, IF IN, OUT, IF IN, OUT, IF IN IN IN IN IN OUT, N OUT, N OUT, N	21 7 7 8	Move Byte, Word, D/Vord, Read Electric Move Pate, Word, D/Vord Swap Dyte: Shift Register Bit Shift Register Bit
MCVD MCVR EMB EMB EMB EMB EMB SMM SMM SRD SRD SLB	IN. COT IPL COT IPL COT IPL COT. II IN. COT. II IN INI COT. N COT. N COT. N	N,	Mare Byte, Word, D'Mard, Ben Block More Fyte, Word, DWord Swap Dyte Shit Right Byte, Word, DWord
MCVD MCVR EMB EM EM EM EM SM SM SM SM SM SM SM SM SM SM SM SM SM	IN, COT IN, COT IN, COT, H IN, COT, H IN Cot, N COT, N COT, N COT, N COT, N	N,N	Move Byte, Word, D'Mord, Read Block Move Fyte, Word D'Word Swap Dyte: Shit Right Byte, Word, D'Mord Shit Right Byte, Word, D'Mord
MCVD MCVR EMB EMM EMB EMM EMB SHM31	IN, COT IN, COT IN, COT IN, COT, H IN, COT, H IN Cots, S., 55 OUT, N COT, N COT, N COT, N COT, N	21 .N	Move Byte, Word, D'Mord, Read Block Move Byte, Word, D'Word Swap Dytes Shift Regist Byte, Mord, D'Mord Shift Luit Byte, Word, D'Mord
MCVD MCVR EMB EMW EMB SWWP SHIST SHI	N. OUT N. OUT N. OUT N. OUT N. OUT N. OUT N OUT N OUT N OUT N OUT N OUT N OUT N	11 [2] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4	Move Byte, Word, D'Alard, Read Elacti Move Pyte, Word, D'Alard Shill Register Bill Shill Register Bill Shill Register Bill Shill Indi Byte, Word, D'Word
MCVD MCVR EMB EMW EMB SMW SMW SMW SMW SMW SMW SMW SMW SMW SMW	N. OUT N. OUT N. OUT N. OUT N N. OUT N N Data S. Di OUT N OUT N OUT N OUT N OUT N OUT N OUT N	11 12 11 11 11	Mare Byte, Word, D'Mard, Ben Black Mare Fyte, Word, DWord Swap Dyte Shit Right Byte, Word, D'Mord Shit Right Byte, Word, D'Mord Shit Right Byte, Word, D'Mord Bactalo Right Byte, Word,
MCVD MCVR EMB EMB EMB SWWP SHB SHW SRD SLE SLE SLE SLE RRD RRD RRD RRD	N. OUT N. OUT N. OUT N. OUT N. OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N	21 21 21 21 21 21 21 21 21 21 21 21 21 2	Move Byte, Word, D'Mord, Real Block Move Fyte, Word Swap Dyte: Shit Register Bit Shit Right Byte, Word, D'Mord Shit Intil Byte, Word, D'Mord Bokate Right Byte, Word, D'Mord
MCVD MCVR EMB EMB EMB SWWP SHR31 SRW SRW SRW SRB SLE SLE SLE SLE RRW RRD RRB RRW RRD CLE SLE	N. OUT N. OUT N. OUT N. OUT N. OUT N. OUT N DUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N	10 10 10 10	Move Byte, Word, D'Mord, Real Block Move Fyte, Word, DWord Swap Dyte: Shift Right Byte, Word, D'Mord Shift Init Byte, Word, D'Mord Shift Init Byte, Word, D'Mord Bootes Right Byte, Word, D'Mord
MCVD MCVD EM EM EM EM EM EM EM SMMP SM SM SM SM SM SM SM SM SM SM SM SM SM	N. GUT N. GUT N. GUT H N. GUT H N. GUT H N N Data, S. Da GUT N GUT N GUT N GUT N GUT N GUT N GUT N GUT N GUT N		Mare Byte, Word, D'Mard, Real Black Mare Byte, Word, D'Word Swap Dytes Shift Right Byte, Word, D'Mard Shift Luit Ryte, Word, D'Mard Blackto Right Dyte, Word, D'Mard D'Mard
MCVD HCVR EMB EMW EMB SWWP SHR31 SR80 SR80 SR80 SR80 SR80 SR80 SR80 SR80	N. OUT N. OUT N. OUT, N. N. OUT, N. N. DIALS, DJ OUT, N. OUT, N. OUT, N. OUT, N. OUT, N. OUT, N. OUT, N. OUT, N. OUT, N.		Mare Byte, Word, D'Mard, Real Block Mare Fyte, Word, Bwap Dyte Shit Right Byte, Mord, D'Mord Shit Right Byte, Word, D'Mord Bolate Right Byte, Word, D'Mord Block Right Byte, Word, D'Mord
MCVD MCVR EME EMW EM EM EM EM EM EM EM EM EM EM EM EM EM	N. OUT N. OUT N. OUT N. N. OUT N N. OUT N N Tala S. DJ OUT N OUT N	13 13 14 15 15 15 15 15 15 15 15 15 15 15 15 15	Mare Byte, Word, D'Mard, Real Block Mane Fyte, Word, Swap Dyte Shit Right Byte, Word, DWord Shit Right Byte, Word, DWord Bokate Right Byte, Word, DWord, DWord Riskate Left Byte, Word, Difford
MCV/D MCV/R EMB EM/W EMB SW/W EMB SW/W SHR31 STR0 STR0 STR0 STR0 STR0 STR0 STR0 STR0	N. OUT N. OUT N. OUT N. OUT N. OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N OUT N	13 13 14 15 15 15 15 15 15 15 15 15 15 15 15 15	Move Byte, Word, D'Mord, Read Block Move Fyte, Word, Swap Dyte: Shift Right Byte, Word, D'Mord Shift Latt Byte, Word, D'Mord Shift Latt Byte, Word, D'Mord Bokate Right Byte, Word, D'Mord Fillana Lett Byte, Word, D'Mord Fill memory space with pattern
MCVD HCVR EME EME EME SHO SHO SHO SHO SHO SHO SHO SHO RR RR RR RR RR RR RR RR RR RR RR RR RR	IN. GUT IN. GUT IN. GUT, H IN. GUT, H IN. GUT, H IN. GUT, H IN. GUT, N GUT, N G		Mare Byte, Word, D'Mard, Real Brock Mare Fyte, Word, DWord Swap Dyte Shift Right Byte, Word, D'Mord Shift Hull Byte, Word, D'Mord Shift Hull Byte, Word, D'Mord Bokete Right Byte, Word, D'Mord Richte Left Byte, Word, D'Mord Fill memory space offic pattern Operations
MAVAD MAZAR EME EME EME EME EME STAN STAD STAD STAD STAD STAD STAD STAD STAD	N. GUT N. GUT N. GUT H N. GUT H N. GUT H N. GUT H N GUT N GUT N	21 ,N 21 21 21	Mare Byte, Word, D'Mard, Real Block Mare Pyte, Word, DWord Swap Dyte Shit Right Byte, Mord, D'Mord Shit Right Byte, Mord, D'Mord Shit Right Byte, Word, D'Mord Boketo, Right Byte, Word, D'Mord Picture Let Byte, Word, D'Mord Fill memory opace with paramy Operations And for cambinations
MAAD MAAD MAAR BAN BAN BAN BAN STAN STAN STAN STAN STAN STAN STAN ST	N. OUT N. OUT N. OUT, H N. OUT, H N. OUT, H N. OUT, N N N N N N N N N N N N N N N N N N N		Mare Byte, Word, D'Mard, Real Block Mane Fyte, Word, DWord Swap Dyte Shill Register Bit Shill Right Byte, Word, DWord Shill hall Byte, Word, DWord DWord Richtle Left Byte, Word, DWord Fill memory space with parton Operations And for combinations Cultors and series
MAAD MAAD MAAD MAAD MAAD MAAD MAAD MAAD	N. CUT N. CUT N. CUT H. N. CUT H N. CUT H N. CUT H N. CUT H N CUT N CUT	3001C	Mare Byte, Word, D'Mord, Real Block More Pyte, Word, DWord Swap Dyte: Shit Right Byte, Word, DWord Shit Left Byte, Word, DWord Shit Left Byte, Word, DWord DWord Riskets Right Byte, Word, DWord Fill memory space with pattern Operation's And for combinations On for combinations
MAND MANE EME EAN EAN EAN EAN SHAR SHAR SHAR SHAR SHAR SHAR SHAR SHAR	IN COUT IN COUT IN COUT IN COUT IN IN COUT IN IN COUT IN COUT IN COUT IN COUT IN COUT	21 ,N 21 21 21 21 21	Mare Byte, Word, D'Mord, Real Book More Pyte, Word, D'Word Sorap Dyte: Shift Right Byte, Word, D'Mord Shift Left Byte, Word, D'Mord Shift Left Byte, Word, D'Mord Bokite, Right Byte, Word, D'Mord Risket Left Byte, Word, D'Mord Fill memory space offic patients Operations And for combinations Operations And for combinations
MAAD MAAD MAAD MAAD MAAD MAAD MAAD MAAD	N. GUT N. GUT N. GUT H N. GUT H N. GUT H N. GUT H N GUT N GUT N GU	21 ,N 21 21 21 21 21 21	Mare Byte, Word, D'Mard, Real Block Mare Pyte, Word, DWord Swap Dyte Shit Right Byte, Word, DWord Shit Right Byte, Word, DWord Shit Right Byte, Word, DWord Bolate Right Byte, Word, DWord DWord Risket Left Byte, Word, DWord DWord Fill memory opace with pattern Operations And for combinations Operations And for combinations Confor combinations
MAAD MAAD MAAR BAB BAB BAB SWAP SHAP SHAP SHAP SHAP SHAP SHAP SHAP SH	N. GUT N. GUT N. GUT, H N. GUT, H N. GUT, H N. GUT, H N The S. F3 OUT, N OUT, N	N N N N N N N N N N N N N N N N N N N	Mare Byte, Word, D'Mard, Real Block Mane Fyte, Word, DWord Swap Dyte Shift Right Dyte, Word, DWord Shift Right Dyte, Word, DWord Shift hall Byte, Word, DWord DWord Richard Left Byte, Word, DWord DWord Differences Logic Panh (datak control) Logic Panh (datak control)
MCAD MCAR EMB EAN EAN EAN EAN EAN SWAP SWAP SWAP SWAP SWAP SWAP SWAP SWAP	N. GUT N. GUT N. GUT N. GUT H N. GUT H N. GUT H N GUT N GUT		Mare Byte, Word, D'Mard, Real Block Mare Pyte, Word, D'Word Swap Dyte Shill Register Bit Shill Register Bit Shill Hell Byte, Word, D'Mord Shill Hell Byte, Word, D'Mord Blatte Right Byte, Word, D'Mord Riante Let Byte, Word, D'Mord Fill memory queue with pattern Operations And to combinations Or for combinations Or for combinations Confor combinations Logic Parth (stack control) Logic Read (stack control) Logic Read (stack control)
MCAD MCAD EMP EAN EAN EAN SWAP SHOT SHOT SHOT SHOT SHOT SHOT SHOT SHOT	IN. GUT IN. GUT IN. GUT IN IN. GUT IN IN. GUT IN IN GUT IN GUT IN		Mare Byte, Word, D'Mard, Real Black Mare Pyte, Word, DWord Swap Dyte Shit Right Byte, Word, D'Mord Shit Hight Byte, Word, D'Mord Shit Hult Byte, Word, D'Mord Bakete Right Byte, Word, D'Mord Riane Lett Byte, Word, D'Mord Fill memory space whit pattern Operations And for combinations or for combinations of for combinations or for combinations of for combinations of for combinations or for combinations
MCVD HCVR EME EME EME EME SWAP STO STO STO STO STO STO STO STO STO STO	N. GUT N. GUT N. GUT H N. GUT H N. GUT H N. GUT H N. GUT H N GUT N GUT N		Mare Byte, Word, D'Mard, Real Block Mare Pyte, Word, DWord Swap Dyte Shit Right Byte, Word, D'Mord Shit Right Byte, Word, D'Mord Shit Right Byte, Word, D'Mord Bolate Right Byte, Word, D'Mord Block Right Byte, Word, D'Mord Block Right Byte, Word, D'Mord Fill memory opace with pattern Operations And for combinations Operations And for combinations Culor combinations Logic Path (datak control) Logic Path (datak control)
MAAD MAAD MAAN BAN BAN BAN BAN SHAP SHAP SHAP SHAP SHAP SHAP SHAP SHAP	N. GUT N. GUT N. GUT H N. GUT H N. GUT H N. GUT H N. GUT N GUT N M		Mare Byte, Word, D'Alard, Real Block Mare Pyte, Word, DWord Swap Dyte Shift Right Byte, Word, D'Alard Shift Right Byte, Word, D'Alard Shift Right Byte, Word, D'Alard Blocks Right Dyte, Word, D'Alard D'Alard Riskets Left Byte, Word, D'Alard D'Alard Logic Panh (stack control) Logic Panh (stack control) Logic Pan (stack control) Logic Pan (stack control)
MCAD MCAD MCAR EMB EXAN END SWAP SWAP SWAP SWAP SWAP SWAP SWAP SWAP	N. GUT N. GUT N. GUT N. GUT N. GUT N. GUT N C C N R N C N C N R N C N R N C N C N		Mare Byte, Word, D'Mard, Real Block Mane Fyte, Word, DWord Swap Dyte Shit Register Bit Shit Right Byte, Word, D'Mord Shit Hitt Byte, Word, D'Mord Blatte Right Byte, Word, D'Mord D'Mard Risten Left Byte, Word, D'Mord Fit memory space with pattern Operation S And for combinations Or for combinations Of for data work off Logic Read (stack control) Logic Peop (stack control) Logic Cord Byte, Word, and
MCAD MCAD MCAR EMP EAN EAN EAN SHOL SHOL SHOL SHOL SHOL SHOL SHOL SHOL	IN. GUT IN. GUT IN. GUT IN. GUT, H IN. GUT, H IN. GUT, H IN. GUT, N GUT,		Mare Byte, Word, D'Mard, Real Black Mare Pyte, Word, DWord Smap Dyte Shit Right Byte, Word, D'Mord Shit Right Byte, Word, D'Mord Shit Latt Byte, Word, D'Mord Riskete Right Byte, Word, D'Mord Riskete Right Byte, Word, D'Mord Fill memory space whit pattern Operations And for combinations of for combinations

XORE N XORW N XORW N XORW N RMB O Table N RMB O Table N RMB O Table N RMB O Table N RMD O Table F FND <strift< td=""> D BCDI C RETI D SEG T NETH N NETH N NETH N HECEF H HECEF H <th>ALOUT ALL AND ALL AND</th><th>Logical XCr of Byte, Ward, or DWard Invest Byte, Word and DWard (I's complement) Inversion Instructions Add data to table Get data from table Find data value in table that matches comparison Convert BCD to but ager Convert BCD to but ager Conve</th></strift<>	ALOUT ALL AND	Logical XCr of Byte, Ward, or DWard Invest Byte, Word and DWard (I's complement) Inversion Instructions Add data to table Get data from table Find data value in table that matches comparison Convert BCD to but ager Convert BCD to but ager Conve
ALTEN N ALTEN N READ N FIND Sale BCDD C BCDD C BCDD C BCDD C BCDI C REATH D BCDI C REATH D BCDI C REATH D SEG T NETH N <td>A, OUT A, OUT A, OUT A, OUT III OT III OT III OT III III</td> <td>DWard Invert Egte, Word and DWard (I's complement) Inversion Instructions Add data to table Get data from table Get data from table Convert BCD to integer Conve</td>	A, OUT A, OUT A, OUT A, OUT III OT III OT III OT III III	DWard Invert Egte, Word and DWard (I's complement) Inversion Instructions Add data to table Get data from table Get data from table Convert BCD to integer Conve
A THU IN A CALL AND A	UT III IIII IIII IIII IIIII IIIII IIIII IIIII IIIIII IIIIIIII	Invent Byte, Word and DWood (1's complement) Add day, to table Got data from table Got data from table Convert BCD to interfer Convert BCD to interfer Decision Free ports contine to model balance parts
IRONA OF INTERNAL	UT U	Invent Byte, Word and DWard (I's complement) Add data to table Get data from table Find data value in table that matches comparison Convert BCD to Integer Convert BCD to Integer BCD to Integer BCD to Integer BCD to Integer BCD to Integer BCD to Integer Alkach Integraph Convert Detach event Straight University Integer BCD to Convert BCD to Integer BCD to Intege
IRAW OR INTERNATION OF TABLE A	UT UT Find, and Co ina, Table 2 adde Data 4 adde data	(I's complement) Neersion Instructions Add day to table Get data from table Find data value in table that natches comparing Convert BCD to interper Convert BCD to interper BCD to interper BCD to interper Convert BCD to interper BCD to interper Convert BCD to interper Convert BCD to interper BCD to interper BCD to interper BCD to interper Convert BCD to interper BCD to interper BC
INTO O TODES. F ATT D LIFO TA FRO TA FND Sa., F FND	AT EVENT	Add data to table Add data to table Get data from table Find data value in table Find data value Fi
Table, F ATT D LIFO T FFO T FFO SC, FND FND FND FND FND FND FND FND FND FND	Ind. and Co and and Co and Data and able Data and able Data and able Data and able Data and able Data and Paten, heds	Add dam to table Add dam to table Got data from table Got data from table Find data value is table that matches comparises Convert BCD to integer Convert BCD to
ATT D LIFO TA FIFO TA FIFO TA FIFO TA FIFO SALE FIND SALE FIND SALE BODI C BODI	Alter Porter and Alter A	Add dam to table Got data from table Got data from table Find data value in table that instales comparises Convert BCD to integer Convert BCD to integer Convert BACD to integer Convert BACD to Red Convert BACD to BAC Convert BACD Con
LIFO T. FIFO T. FIFO T. FIFO T. FIND S FIND S	Adde Data I adde Data I adar, Inda I Patro, Inda I N, OUT I I N, OUT I I I N, OUT I I N, OUT I I I I I I I I I I I I I I	Get data from table Find data value in table Find data value in table Corvert BCD to be user Corvert BCD to be user Corvert BCD to be user Corvert BCD to BCA Corvert BCD Corvert
FIFO T FND SC. FND	Ade Data di ann, Inda di Patra, Inda di Patra, Inda di Patra, Inda di UT Aut N, OUT di N, O	Find data value is table that matches comparison Convert BCD to integer Convert BCD to integer Convert Integer to 10.0 Convert Peestro Nord Convert Peestro Nord Convert Restro Ascill Convert Restro Ascill Convert Restro Ascill Decide Eacode Convert Ascill to HEX Convert Restro Ascill Decide Eacode Convert Ascill to HEX Conditional Februaries Astrono Free Interrupt Conditional Februaries Disable Interrupt Conditional Interrupt Eacode Interrupt Conditional Interrupt Eacode Interrupt Conditional Interrupt Condition Interrupt Attack Interrupt Decide Interrupt Condition Decide Interrupt Condition Freeport Units Time Interrupt
FND-Sc. FND-Sc	Autro, Inde de Patre, Inde de Patre, Inde de Patre, Inde de Patre, Poetre de Patre, Poetre, Poetre de Patre, Poetre de P	Find data value in table that matches comparison Convert BCD to interper Convert BCD to interper Convert ErAbard to Real Convert ErAbard to Real Convert Fieshto DWard Convert Real to HEX Convert Real to ABCD Convert Real to ABCD Convert Real to ABCD Convert Real to ABCD Convert Real to ABCD Decode Encode Convert Real to ABCD Encode Convert Real to ABCD Encode Conv
FND Sar, FND Sar, FND Sar, FND Sar, FFND Sar, FFDD Sar,	Pains, Indx II Pains, Indx II Pains, Indx II Pains, Indx II VUF V, OUT II N,	Find share values in table that naticless comparisons Convert BCD to be seen per Convert BCD to be seen per Convert Books DMard Convert Books DMard Convert Books DMard Convert BCD to HEX Convert BCD to HEX Decode Encode Convert BCD to HEX Decode Encode Convert BCD to HEX Convert BCD to HEX Convert BCD to HEX Convert BCD to HEX Convert BCD to HEX Direct BCD to HEX Direct Convert BCD Freeport to Convert BCD
FND=SELF FND=SELF FND=SELF BCD1 C BCD1 C BCD	Pain, liedx a Pain, liedx a UP Aur VUP VUP VUT V, OUT a V, OUT A V	matches companies Convert BCD to integer Convert BCD to integer Convert Boosto DWord Convert Resto DWord Convert Resto DWord Convert Resto DWord Convert REX to ABCH Decode Convert REX
FND> Sar, F BCDI C BCDI C BCD C BCD C BCD C BCD C BCD C BCC C C C C	Patin, fields III UT UT N, OUT N, OUT	Convert BCD to Interpret Convert Integrate II.30 Convert Integrate II.30 Convert Boolto Different Convert Boolto Different Convert ASCII to HEX Convert HEX to ASICII Decode Encode Convert HEX to ASICII Decode Convert HEX to ASICII Convert HEX to ASICII Decode Convert HEX to ASICII Convert HEX to ASICII Decode Convert HEX to ASICII Convert HEX to ASICII Decode Convert HEX to ASICII Convert HEX to ASICII Convert HEX to ASICII Decode Convert HEX to ASICII Convert HEX t
BCDI C BCDI C BCD C IBCD C IBCD C IBCD C IBCD C IBCC C	NUF AL OUT AL OUT AL OUT AL AL OUT AL AL OUT AL AL OUT AL	Convert BCD to latence Convert Integer to 1577 Convert Integer to 1577 Convert Readto DMard Convert Readto DMard Convert Readto DMard Convert Readto DMard Convert Readto DMard Convert Readto DMard Encode Encode Convert Readto DMard Encode Convert Readto DMard Convert Readto DMard Convert Readto DMard Convert Readto DMard Convert Readto DMard Detects essent Convert Readto DMARD Detects essent Convert Readto DMARD Detects essent Convert Readto DMARD Freeport excluse readto DMARD
HICEF I	AUT V, OUT V, OUT V	Convert Integer to 1020 Convert Presito DWard Convert Resito DWard Convert Resito DWard Convert Resito DWard Convert REX to Astall Decode Encode Converts 7 segment potter Property Conditional Feters from Interrupt Return from Interrupt Conditional Feters from Interrupt Return from Interrupt Disable Interrupts Disable Interrupt contine to ment Detach second Detach second Conditional Freeport to consecution for Freeport to consecution for
DIR I I TIRUNG I ATH I DECO I ENCO I SEG. I INT N CREIL RETI ENI DESI ENI DESI ENI DESI ENI DESI ENI DESI ENI DESI I NETR I NETR	N, OUT 3, V, OUT 3, V, OUT 3, V, OUT 1, EN 1, N, OUT 1, EN 1, N, OUT 1, OUT 1, N, OUT	Convert ErAnd to Real Convert Feeshe Divard Convert Resite Divard Convert Resite Allow Encode Convert HEX to Allow Encode Construct 7 segment pottern errupt Engineer of Interrupt resting Conditional Februs form Interrupt Return from Interrupt Divado Interrupt
TRUNG P ATH P ATH P DECO P ENICO P ENI	N, GUT 14 N, GUT EN N, GUT EN N, GUT N, GUT N, GUT Int A NT, EVENT MILE, PORT MALE, PORT MALE, PORT	Convert Realto DWard Convert RECHIS (IEX Convert RECHIS (IEX Convert RECHIS (IEX) Decode Convert RECHIS (IEX) Converts 7: segment potter orrupt Converts 7: segment potter orrupt Converts (Converts) Converts (Converts) Converts) Converts (Converts) Converts) Converts
ATH PAAN PANA PANA PANA PANA PANA PANA PAN	N, OUT, LEN N, OUT, LEN N, OUT N, OUT N, OUT Int N, OUT Int N, EVENT EVENT EVENT EVENT EVENT EVENT EVENT	Consent ASCII to HEX Consent HEX to ASCII Decode Encode Consents 7 acqueent potter off Upt Designing of Interrupt rectine Constitution from form Interrupt Return from Interrupt Dis Aldo Interrupt Dis Aldo Interrupt Dis Aldo Interrupt Dis Aldo Interrupt Decode event United tion Freeport transmission Freeport to clice managem
HTA P DECO P ENCO P SEG P INT N ORET RET ENI DIS ATCH I DIS ATCH I DIS ATCH I NETR I NETR I NETR I HDEF I HDEF I	N, OUT, LEN N, OUT N, OUT N, OUT N, OUT Int N, EVENT EVENT EVENT EVENT EVENT N, EVENT N, EVENT N, EVENT	Convert HEX to Asicit Decode Encode Contracto 7: acquirent pottern orrupt Techning of Interrupt restine Conditional Feterin form Interrupt Return from Interrupt Conditional Interrupt Disable Interrupt Absolute Interrupt Absolute Interrupt Detachement Detachement Stratic ation Freeport excess resting on
DECO I ENCO I SEG. I INT N GRER RETI ENI DESI ATCH I DICH E NETR I NETR I NETR I NETR I NETR I NETR I NETR I NETR I NETR I	N, OUT N, OUT N, OUT Int Int Int EVENT EVENT EVENT EVENT EVENT EVENT INT INT INT INT INT INT INT INT INT I	Decode Encode Contento 7 segment potters OTTUPL Despinsing of Interrupt resting Conditional Return form Interrupt Return from Interrupt Disable Interrupts Disable Interrupt Attach Interrupt contine To- ment Detects event String of On Freeport transmission Freeport transmission
ENCO P SEG T INT N CRER RETI ENI DISI ATCH I DTGH E XUAT T RCV I NETR I NETR I NETR I NETR I NETR I NETR I NETR I	N, OUT N, OUT Int Int INT, EVI-NT MILE, NT, EVI-NT ANLE, NT, 2 NALE, NT, 2 NALE, PORT	Encode Constals 7 segment potent orrupt Depinning of Interrupt restin Conditional Fourier form Interrupt Return from Interrupt Dis Jobs Interrupt Dis Jobs Interrupt Attach Interrupt content to mend Detach ment Unic attion Freeport transmission Freeport to clice the one of the
SEG I INT N DREN RETI ENI DBS ATCH I DTGH E XANT 1 RCV 1 NETR 1 N	N, OUT	Constain 7 ac grand pattern arrupt Constituent for the rupt restin Constituent for the rupt Return from Interrupt Bit die Interrupt Dit die Interrupt Altroch Interrupt Action Detach ment Durié ation Freeport transmittion Freeport to chie ment an
INT N CREEN RETT ENI DISH ATCH I DTCH E NCT T NETR T NETR T NETW 1 HDEF I HDEF I HSC 1	Int NIT, EVENT EVENT Comm BALL, PORT NALE, EVENT ALLE, CALL, 2 NALE, PORT ALLE, 2 NALE, 2 NALE	errupt Engines of Interrupt restin Conditional Felanic Form Interrupt Return from Interrupt English Interrupt Dis Alls Interrupt Allsch Interrupt routine To- ment Detach event Sunic ation Freeport university for Freeport to clice management
INT N INT N IN	NT, EVENT EVENT Comm NATE PORT AGLE, CRT 2 CRT 2	Terreport Terreport Constituent Constituent Interrupt Return from Interrupt Return from Interrupt Dis Julio Interrupt Dis Julio Interrupt Attach Interrupt Detach event Stradio Tion Freeport to clice managem
INT NERT I RETI ENI DIGH E XATCH I DIGH E XATCH I NETR I NETR I NETR I NETR I NETR I NETR I NETR I	NT, EVENT Comm Date Post Aste S RT 2	Conditional Return from Interrupt Return from Interrupt Earth Interrupt Disable Interrupts Disable Interrupt Attach Interrupt contine to most Detach second Unite attion Freeport transmission Freeport transmission
CHEETI RETI ENI DISI ATCH I DTGH E XUAT T RCV I NETR I NETR I NETR I NETR I NETR I NETR I NETR I NETR I	NT, EVENT Comm Date post Aste s. RT 2 Date p. RT	Constitution result (2011) Interrupt Return from Interrupt Early Interrupts Dis Jobs Interrupts Attroch Interrupt (2014) Delach ment Delach ment Unic ation Freeport transmission Freeport to case and an
ENI ENI DBA ATCH I DTGH E XMIT 1 RCV 1 NETR 1 NETR 1 NETR 1 NETR 1 NETR 1 NETR 1 NETR 1 NETR 1 NETR 1	NT, EVENT EM NE Comm Indele, port Indele, NE A	Return from Interrupt Early Interrupts Dialade Interrupt contine to main Detachement Unicotion Freeport transmission Freeport transmission
ENI DISI ATCH I DTGH E XMIT 1 RCV 1 NETR 1 NETW 1 HDEF I HDEF I HSC 1 FLS 2	NT, EVENT Comm Date port IABLE, IABLE P. RT 1	Earth and an up to Dis data Interrupt continue to many Detects event Control of the terrupt Control of the terrupt Freeport terrupt terrupt and Freeport terrupt terrupt
DISH ATCH I DTCH E XANT T RCV 1 NETR 1 NETR 1 NETR 1 NETW 1 HCEF I HSC 1 FLS 2	NT, EVENT Comm DALL, PORT IABLE, IABLE, RT	Bi dia hiternapi Attach hiternapi contine to mul Detach esent sunication Freeport transmission Freeport to circo mucoun
ATCH I DTGH E XMIT 1 RCV 1 NETR 1 NETW 1 HCEF I HSC 1 PLS 2	NT, EVENT Comm Date Port Inste	Altach Interrupt routine to menti Detach event sunication Freeport transmission Freeport to circo name on
ALEAN DTGH E XLAT 1 RCV 1 NETR 1 NETR 1 NETW 1 HDEF 1 HBC 1 PLS 2	Comm Comm Init, Port IAELE, IAELE, RT 2	numi Detach event which tion Freeport transmission Freeport (sociale transmission
XIAI I RCV I NETR I NETW I HDEF I HBC I PLS 2	Comm DATLE PORT IABLE, IABLE PORT	Detach event sunication Freeport transmission Freeport to cale the end
XMIT 1 RCV 1 NETR 1 NETW 1 HOEF 1 HOEF 1 HOEF 1 HOEF 1	Comm Infle, Port IABLE, INF 2 IABLE, PORT 2	NUMICATION Freeport Units and mich Freeport to sale turn of m
XMT 1 RCV 1 NETR 1 NETW 1 HCEF 1 HSC 1 PLS 2	TABLE, PORT	Freeport Uses and inten Freeport (see in elements in
RCV T NETR T NETW T HCEF H HBC N PLS 2	IAELE.	Freeport escale terminaria
NETR 1 NETW 1 HOLEF 1 HINC 1 PLS 2	TABLE PORT	Linedow Line of the solution
NETR 1 NETW 1 HDEF 1 HBC 1 FLS 2	ABLEPRI	Contraction of the second s
NETW 1 HOEF H HSC 1 PLS 2	the second second	Hetwark Fizad
HDEF I	ABLE PORT A	Hetwoolk White
HCEF H HBC I PLS 2	High Spee	dinstructions
HINC I	-Rei, Mode	Beline High Speed Counter
HBC I PLS 2		INCOLO
PLS J	N	Activate High Speed Counte
	X II	Palar Culpul
Instruct S7-00 foll win	tions are val PLCs a= ma ng key: 214, 215, am 215, and 21 marked, the i	id for the individual rk: d a cording to the d 216 only only nstructions ar valid

2. PROCESS

2.1. Introduction

There is a need for a program for the PLC controller. STEP-7 Microwin PLC is a program editor prepared for Siemens PLC. The same program editor will be used for my research purposes. The unit that will be controlled by the Siemens S-7 CPU 212 PLC is a dry packaging unit. This can be seen in the fig.3. The flowchart for the PLC program can be seen fig.4.



Fig.3 dry packaging unit



Fig. 4 PLC program flowchart

The program of the PLC and the series port allows for outside communication. Communication between the PLC and series port needs codification and de-codification systems. Apart from this there is a need for a software for the PLC. However the aim of the project is to find a simpler system of communication between the computer and the series port.

One of the aims is to prepare the PLC to control the input contacts and run the on/off functions. The incoming output contacts of the PLC can be read by the help of the parallel ports. There are two important steps in this process:

- I. A program is needed for the control of the parallel ports.
- II. An electronic circuit is needed for the control of the information received from the
- parallel ports.

2.2. Parallel Port Controlling program

This program is written by using Borland Delphi 5. the program was designed as shown to give easy recognition and be easily understood. As it can be seen from the figure this system can make 8 output controls. The program also manages the 4 inputs as well as the external information income. Information are coded is binaries. Fault warnings are displayed at the upper bar of the program.

Two PC can easily be connected to communicate with each other by using one as the host. In order to communicate from a distance a program called "PC anywhere" can be used.

2.3. Electronic Circuit

The circuit design can be seen in the figure. There are further explanations about the circuit. The parallel port control circuit (PPCC) enables computer and PLC connection without an electric connection. The system contains optocopler and relays.

2.4. Server Environment

Server environment encompasses a PC-Anywhere program which can initiate communication by warning the PC via a modem which is programmed earlier. When the connection is made one of the devises act as server and the other as client.

- I. Server computer is the computer that manage the client computer from a distance.
- II. Client computer is the computer that had been targeted by the server computer.

Two computers can be connected in the following fashion:

- I. Direct
- II. Via a Modem
- III. Network
- IV. Internet
 - I. Direct

Two computers can be connected directly by a UTP cable. An ethernet card is needed for the connection. Such a system can be adapted for the control of packaging from within the building.

II. Modem

A telephone line can be used for a modem connection. By using PC anywhere program it is possible to communicate with the client computer from anywhere. All we need is a computer, a telephone line and a modem. Distance is no problem with this system.

III. Network

some large organization can setup their own network systems. PC anywhere can be used for these systems as well.

IV. Internet

IP number of the client computer needs to be known to establish a connection via an internet provider. PC anywhere program can be connected to the internet and by registering an IP number the connection can be made.

2.5. Communication and Client Environment

Client environment consist of a computer, PPCD, pre-installed PC anywhere modem and the telephone line. Apart from these the program I wrote with the Delphi 5 is needed to control the PPCD. Since the Delphi Assembly language is easily acceptable it is much easier to make the parallel port control. Step-7 microwin program is needed if there is a desire to introduce other programs in future. Program editor is providing clear view of the steps of actions. Client can communicate with more than one servers shown fig.5. If desired the server can communicate with more than one client shown fig.6.



Fig.5 Client can view more than one process.



Fig.6 Process can be viewed by more than one client.

2.6. Connection for Chat

Need may arise to send and receive messages during the communication. A part is designed with Delphi which can be used as a chat shown fig.7.



Fig.7 Message line for chat

2.7. Process and Alarm

The system provides information about the progress of the work. System can be improved with extra functions. For example, it can be arranged that the system can send sms messages to our mobile telephones in case of emergencies. The program can be improved to include the capabilities of sms communication. Prewritten statements can then be sent to the mobile telephone. In order to achieve this the information at the outputs need to be transferred to the 4 bit inputs, and the binaries taken from here should be evaluated as hexadecimal. The incoming information are registered at H379 address. The control program of the computer then put these in order. Here, the Q0.0, Q0.1, Q0.2, Q0.3, Q0.4, output are controlled. All the output is send one after the other. Messages will reach us in case there is a problem with the computer itself. Any correction, amendment and configuration shall be possible to be made from the distance.

3. PRODUCTION UNIT

3.1. Introduction to the Production Unit

Dry packaging unit consist of several units as it can be seen in the figure 8. The system is controlled by relays. This is a system that requires extensive maintenance work. The system can be modified to be controlled by PLC system.





Fig.9 Back side of the productions unit

3.2. Air ventilation fan

The fun functions to clean the dust from the goods that will be packaged. It consist a 3Hp 415 V 50Hz motor. According to regulations in TRNC a delta starter switch needs to be used for motors between 3Hp and 5.5HP.

3.3. Silo 1.

The first silo is where the goods are initially stored. A pallet is used for pouring the good into the silo. The silo is hand fed.

3.4. Silo 2.

After the first silo is full the second is started to be filled. There are switches that triggers the pallet at each silo. As it can be seen at the photos the two silos have height adjustments. The heights are arranged by a push button. There are also adjustments for the weight of the goods that will be packaged. Silo 1. and silo 2. are also facilitated with limit switch which control the discharge of the silos.

3.5. Nylon bag production unit

An electric and a mechanical system is to run nylon film and weld it to become a bag. The nylon tube has black markings, which are recognized by the sensors and these give the orders for action to seal the nylon. Figure 1.1



Fig.1.1 Nylon bag production unit

3.6. Sensors

The system contains 2 level switch and 1 limit switch. Role of these were just explained above. The sensors are made from photo-led and transistors. Shown fig.1.2a and 1.2b



Fig.1.2a Black band reading sensor, Fig1.2b limit switch

4. PORT CONTROL UNIT

4.1. Parallel Port Control

There is a 25 pin D type parallel port on the computer. These can be used for sending information from the computer (output). It is also possible to obtain information from outside and feed these to the computer from outside using the pins (input). It's shown fig1.3.



Fig.1.3 Parallel ports input and output

Standard signals sent and received from these ports are at the standard TTL level. As it can be understood the information exchange is made a by a parallel line the system allows more than one parallel port. These ports can be named as LPT1, LPT2, and LPT3. The IBM 25 D type connector numbering can be seen at the figure.

4.2. Port Register

In order to obtain data exchange the data need to be registered somewhere when the computer is connected to a parallel port. Data registration is temporarily made at the register section. Each register is represented by a hexadecimal address. 378H, 379H are such register addresses. As it can be seen the inputs are represented by S and the output are represented as D bites.

4.3. Parallel Port Control Circuit (PPCC)

Each bit taken from the parallel port is known to be at TTL level 5V. When a current connected to the system exceeds 1mA the computer can be damaged. In order to avoid such problem a circuit needs to be designed. It is shown figure 1.4. The circuit should enable the controls to be made without any electrical connections. An optocopler system can be used for this purpose. Optocopler is a system which consist a led and photo transistors. This enables a secure and safe control. BC 148 or 2N2222 is activated by the information received from the computer. When the BC 148 is activated it activates the collector and the emitter which then energizes the relay.



Fig.1.4 Port control circuits block diagram

The 470 Ohms resistance between the optocopler and the Parallel port is safeguard for the led. Led has a feature of 1.7 V. 470 Ohms resistance that is connected to the base of the Transistor limits the 0.7 V, which is necessary to activate it. Relay is activated by 25 V DC voltage, which then activates the PLC inputs. PLC outputs are communicated to the computer by the relay contacts. When the inputs are in connection 4 bit input is at the 0000 position. Energized relay opens the closed contacts and the output and input are evaluated with the program designed with Delphi.



Fig.1.5a PPCC



Fig.1.5b PPCC' printed board

I made the drawings for the PPCC with a program called Board Maker. then I made a printed board of the design. The circuit and the assembly can be seen fig.1.5a and b

4.4. Parallel Port Control Program

I selected Borland Delphi 5 for programming fig1.6. This is because it is very adequate to make a good visual design with this program. This program is perfectly in harmony with the assembly language. Assembly language is necessary for the port control. There is a necessary program under each button. The whole program is listed below. As it can be seen in the figure the program is designed for general purposes. The outputs can be controlled individually. There is also a section for the hexadecimal display. The inputs can individually be displayed as binaries as well as hexadecimal. There is a row for a message. The information from the inputs is displayed as messages at this row.
Port Control							×
for Single System OUTPUTS							
Start Stop		0 0	linare Value	Of Outouts	Data		
						01	DO
	D7 D	06 05	U4	03	UZ		
Number DI Pakage	STABL		Decimal	Value D		Dete	
Count Up Count Down	RESET	-	Decimo	Talat C			
Reset	STOP						
Heater Times							
Count Up Reset START STOP							
System to On Position				Jecimal	value U	-	
) (0	0	1
CLEAR TEXT							

Fig.1.6 program of part control

ъ

Borland Delphi 5 PC's port control program

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons, ExtCtrls;

type

TForm1 = class(TForm) GroupBox1: TGroupBox; Button3: TButton; Button4: TButton; Button5: TButton; Button6: TButton; Button7: TButton; Button8: TButton; Button9: TButton; Button10: TButton; Edit2: TEdit; Edit3: TEdit; Edit4: TEdit; Edit5: TEdit; Edit6: TEdit; Edit7: TEdit; Edit8: TEdit; Edit9: TEdit; Edit14: TEdit; GroupBox2: TGroupBox; ComboBox1: TComboBox; Button1: TButton; Edit1: TEdit; Button2: TButton; Label1: TLabel; Label2: TLabel; GroupBox3: TGroupBox; Button14: TButton; Button16: TButton; Edit15: TEdit; Button17: TButton; Label4: TLabel; GroupBox4: TGroupBox; Button15: TButton; Edit16: TEdit: Button18: TButton; Label5: TLabel; GroupBox5: TGroupBox; Edit10: TEdit; Button11: TButton; Edit12: TEdit; Button13: TButton; Button12: TButton; Edit11: TEdit; Label3: TLabel; Edit13: TEdit; GroupBox6: TGroupBox; Button19: TButton; Edit17: TEdit; Button20: TButton; Edit18: TEdit;

Label6: TLabel; GroupBox7: TGroupBox; Edit19: TEdit; Timer1: TTimer; Led1: TShape; Led2: TShape; Led3: TShape; Led4: TShape; Label7: TLabel; Label8: TLabel; Label9: TLabel; Label10: TLabel; Label11: TLabel; Label12: TLabel; Label13: TLabel; Label14: TLabel; Label15: TLabel; Memo1: TMemo;

SpeedButton1: TSpeedButton;

procedure Button1Click(Sender: TObject); procedure Button2Click(Sender: TObject); procedure FormCreate(Sender: TObject); procedure Button11Click(Sender: TObject); procedure Button3Click(Sender: TObject); procedure Button4Click(Sender: TObject); procedure Button5Click(Sender: TObject); procedure Button6Click(Sender: TObject); procedure Button7Click(Sender: TObject); procedure Button8Click(Sender: TObject); procedure Button9Click(Sender: TObject); procedure Button12Click(Sender: TObject);

procedure Button13Click(Sender: TObject);

procedure Button14MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button16MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button17MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button15MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button18MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button14MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button16MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button17MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button15MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button18MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button19MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button19MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button20MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button20MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button3MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Button13MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button21Click(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

procedure Button22MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button22MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button23MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button24MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button25MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button23MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button24MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure Button25MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

procedure SpeedButton1Click(Sender: TObject);

procedure GroupBox2Click(Sender: TObject);

private

{ Private declarations } public

D.1.12 1 1

{ Public declarations }

end;

var

Form1: TForm1;

Veri:Integer; D0:Integer; D1:Integer; D2:Integer; D3:Integer; D4:Integer; D5:Integer; D6:Integer; D7:Integer; Cup:Integer; Toplam:Integer; Up:Integer; sonuc:Integer; implementation

{\$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject); begin if not (combobox1.text=") then begin GroupBox1.Enabled:=False; GroupBox3.Enabled:=False; GroupBox4.Enabled:=False; GroupBox6.Enabled:=False; edit1.color:=clGreen; Button2.Enabled:=True; If Combobox1.Text=('Device1') Then veri:=1; If Combobox1.Text=('Device2') Then veri:=2; If Combobox1.Text=('Device3') Then veri:=4; If Combobox1.Text=('Device4') Then veri:=8;

If Combobox1.Text=('Device5') Then veri:=16; If Combobox1.Text=('Device6') Then veri:=32; If Combobox1.Text=('Device7') Then veri:=64; If Combobox1.Text=('Device8') Then veri:=128; asm mov al, veri mov dx,378h out dx,al end; Button1.Enabled:=False; Label1.Visible:=True; //Combobox1.enabled:=False; end else showmessage('Please select any device'); combobox1.setfocus; end: procedure TForm1.Button2Click(Sender: TObject); begin GroupBox1.Enabled:=True; GroupBox3.Enabled:=True; GroupBox4.Enabled:=True; GroupBox6.Enabled=True; Button19.Enabled:=True; Button20.Enabled:=True; edit1.color:=clRed; Button1.Enabled := True; Button2.Enabled:=False; Label1.Visible:=False; Combobox1.Text:="; //ComboBox1.Enabled:=True;

asm mov al,0 mov dx,378h out dx,al end; end;

procedure TForm1.FormCreate(Sender: TObject); begin D7:=1; D6:=2; D5:=4; D4:=8; D3:=16; D2:=32; D1:=64; D0:=128; Edit15.Text:="; Edit15.Text:="; Up:=0; CUp:=0; end;

procedure TForm1.Button11Click(Sender: TObject); begin If Not (Edit2.Color=clGreen) Then D0:=0; If Not (Edit3.Color=clGreen) Then D1:=0; If Not (Edit4.Color=clGreen) Then D2:=0; If Not (Edit5.Color=clGreen) Then D3:=0; begin

Edit3.Text:=' 1'; Edit3.color:=clGreen; Button11.Enabled:=True; Button13.Enabled:=True; Edit12.color:=clYellow; Button4.enabled:=False; end;

procedure TForm1.Button5Click(Sender: TObject); begin Edit4.Text:=' 1'; Edit4.color:=clGreen; Button11.Enabled:=True; Button13.Enabled:=True; Edit12.color:=clYellow; Button5.enabled:=False; end;

procedure TForm1.Button6Click(Sender: TObject); begin Edit5.Text:=' 1'; Edit5.color:=clGreen; Button11.Enabled:=True; Button13.Enabled:=True; Edit12.color:=clYellow; Button6.enabled:=False; end;

procedure TForm1.Button7Click(Sender: TObject); begin Edit6.Text:=' 1'; Edit6.color:=clGreen; Button11.Enabled:=True; Button13.Enabled:=True; Edit12.color:=clYellow; Button7.enabled:=False; end;

procedure TForm1.Button8Click(Sender: TObject); begin Edit7.Text:=' 1';

Edit7.color:=clGreen; Button11.Enabled:=True; Button13.Enabled:=True; Edit12.color:=clYellow; Button8.enabled:=False; end;

procedure TForm1.Button9Click(Sender: TObject); begin Edit8.Text:=' 1'; Edit8.color:=clGreen; Button11.Enabled:=True; Button13.Enabled:=True; Edit12.color:=clYellow; Button9.enabled:=False; end;

procedure TForm1.Button10Click(Sender: TObject); begin Edit9.Text:=' 1'; Edit9.color:=clGreen; Button11.Enabled:=True; Button13.Enabled:=True; Edit12.color:=clYellow; Button10.enabled:=False; end;

procedure TForm1.Button12Click(Sender: TObject); begin D7:=1; D6:=2; D5:=4; D4:=8; D3:=16; D2:=32; D1:=64; D0:=128; Edit2.Text:=' 0'; Edit3.Text:=' 0'; Edit4.Text:=' 0'; Edit5.Text:=' 0'; Edit6.Text:=' 0'; Edit7.Text:=' 0';Edit8.Text:=' 0'; Edit9.Text:=' 0'; Button3.Enabled:=True; Button4.Enabled:=True; Button5.Enabled:=True; Button6.Enabled:=True; Button7.Enabled:=True; Button8.Enabled:=True;

Button9.Enabled:=True; Button10.Enabled:=True; Edit2.Color:=clRed; Edit3.Color:=clRed; Edit4.Color:=clRed; Edit5.Color:=clRed; Edit6.Color:=clRed; Edit7.Color:=clRed; Edit8.Color:=clRed; Edit9.Color:=clRed; Edit10.Color:=clRed; Edit11.Color:=clRed; Edit13.Text:="; asm mov al,0 mov dx,378h out dx,al end; Button12.Enabled:=False; GroupBox1.Enabled:=True; //Button11.Enabled:=True; end;

procedure TForm1.Button13Click(Sender: TObject); begin D7:=1; D6:=2; D5:=4; D4:=8; D3:=16; D2:=32;

41

D1:=64; D0:=128; Edit2.Text:=' 0'; Edit3.Text:=' 0'; Edit4.Text:=' 0'; Edit5.Text:=' 0'; Edit6.Text:=' 0'; Edit7.Text:=' 0'; Edit8.Text:=' 0'; Edit9.Text:=' 0'; Button3.Enabled:=True; Button4.Enabled = True; Button5.Enabled:=True; Button6.Enabled:=True; Button7.Enabled:=True; Button8.Enabled:=True; Button9.Enabled:=True; Button10.Enabled:=True; Edit2.Color:=clRed; Edit3.Color:=clRed; Edit4.Color:=clRed; Edit5.Color:=clRed; Edit6.Color:=clRed; Edit7.Color:=clRed; Edit8.Color=clRed; Edit9.Color:=clRed; Edit12.color:=clRed; Button11.Enabled:=False; asm mov al,0 mov dx,378h

out dx,al end; Button13.Enabled:=False; end;

procedure TForm1.Button14MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin if (Up=100) Then Up:=-1; Up:=Up+1; Edit15.Text:=inttostr(Up); asm mov al,128 mov dx,378h out dx,al end; end;

procedure TForm1.Button16MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin if (Up=0) Then Up:=101; Up:=Up-1; Edit15.Text:=inttostr(Up); asm mov al,64

```
mov dx,378h
out dx,al
end;
end;
```

procedure TForm1.Button17MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin Up:=0; Edit15.Text:=inttostr(Up); asm mov al,192 mov dx,378h out dx,al end; end;

procedure TForm1.Button15MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin CUp:=CUp+1; if (CUp=11) Then CUp:=0; Edit16.Text:=inttostr(CUp); asm mov al,32 mov dx,378h out dx,al end; end; procedure TForm1.Button18MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin Edit16.Text:='0'; CUp:=0; asm mov al,16 mov dx,378h out dx,al end; end;

procedure TForm1.Button14MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end;

procedure TForm1.Button16MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end; end;

procedure TForm1.Button17MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end; end;

procedure TForm1.Button15MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end;

procedure TForm1.Button18MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm

mov al,0

mov dx,378h

out dx,al

end;

end;

procedure TForm1.Button19MouseDown(Sender: TObject; Button:

TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

Edit17.Color:=clGreen;

asm

mov al,1

mov dx,378h

out dx,al

end;

end;

procedure TForm1.Button19MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin Edit17.Color:=clRed; asm mov al,0 mov dx,378h out dx,al end; end; procedure TForm1.Button20MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin Edit18.Color:=clGreen; asm mov al,2 mov dx,378h out dx,al end; end;

procedure TForm1.Button20MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin Edit18.Color:=clRed; asm mov al,0 mov dx,378h out dx,al end; end;

procedure TForm1.Button3MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin GroupBox6.Enabled:=False; GroupBox2.Enabled:=False; GroupBox3.Enabled:=False; GroupBox4.Enabled:=False; end;

procedure TForm1.Button13MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin GroupBox6.Enabled:=True; GroupBox2.Enabled:=True; GroupBox3.Enabled:=True; end;

procedure TForm1.Button21Click(Sender: TObject); begin asm mov dx,379h in al,dx xor al,128 //mov cl,3 //shr al,cl and al,11110000b mov sonuc,al end; Edit19.Text:=InttoStr(sonuc); end;

procedure TForm1.Timer1Timer(Sender: TObject); Const x:integer=0;

begin asm mov dx,379h in al,dx xor al,128 //mov cl,3 //shr al,cl and al,11110000b mov sonuc,al end; Edit19.Text:=InttoStr(sonuc); If Edit19.Text=('0') Then Begin Led1.Brush.color:=ClWhite; Led2.Brush.color:=ClWhite; Led3.Brush.color:=ClWhite; Led4.Brush.color:=ClWhite; Label7.Caption:='0'; Label8.Caption:='0'; Label9.Caption:='0'; Label10.Caption:='0'; Label12.Caption:='0'; end; If Edit19.Text=('16') Then Begin Led1.Brush.color:=ClRed; Led2.Brush.color:=ClWhite; Led3.Brush.color:=ClWhite; Led4.Brush.color:=ClWhite; Label7.Caption:='1'; Label8.Caption:='0';

Label9.Caption:='0'; Label10.Caption:='0'; Label12.Caption:='1'; end; If Edit19.Text=('32') Then Begin Led1.Brush.color:=ClWhite; Led2.Brush.color:=ClRed; Led3.Brush.color:=ClWhite; Led4.Brush.color:=ClWhite; Label7.Caption:='0'; Label8.Caption:='1'; Label9.Caption:='0'; Label10.Caption:='0'; Label12.Caption:='2'; end; If Edit19.Text=('48') Then Begin Led1.Brush.color:=ClRed; Led2.Brush.color:=ClRed; Led3.Brush.color:=ClWhite; Led4.Brush.color:=ClWhite; Label7.Caption:='1'; Label8.Caption:='1'; Label9.Caption:='0'; Label10.Caption:='0'; Label12.Caption:='3'; end; If Edit19.Text=('128') Then Begin

Led1.Brush.color:=ClWhite;

Led2.Brush.color:=ClWhite; Led3.Brush.color:=ClRed; Led4.Brush.color:=ClWhite; Label7.Caption:='0'; Label8.Caption:='0'; Label9.Caption:='1'; Label10.Caption:='0'; Label12.Caption:='4'; end; If Edit19.Text=('144') Then Begin Led1.Brush.color:=ClRed; Led2.Brush.color:=ClWhite; Led3.Brush.color:=ClRed; Led4.Brush.color:=ClWhite; Label7.Caption:='1'; Label8.Caption:='0'; Label9.Caption:='1'; Label10.Caption:='0'; Label12.Caption:='5'; end; If Edit19.Text=('160') Then Begin . Led1.Brush.color:=ClWhite; Led2.Brush.color:=ClRed; Led3.Brush.color:=ClRed; Led4.Brush.color:=ClWhite; Label7.Caption:='0'; Label8.Caption:='1'; Label9.Caption:='1'; Label10.Caption:='0';

Label12.Caption:='6'; end;

If Edit19.Text=('176') Then Begin

Led1.Brush.color:=ClRed;

Led2.Brush.color:=ClRed;

Led3.Brush.color:=ClRed;

Led4.Brush.color:=ClWhite;

Label7.Caption:='1';

Label8.Caption:='1';

Label9.Caption:='1';

Label10.Caption:='0';

Label12.Caption:='7'; end;

If Edit19.Text=('64') Then

Begin

Led1.Brush.color:=ClWhite;

Led2.Brush.color:=ClWhite; Led3.Brush.color:=ClWhite;

Led4.Brush.color:=ClRed;

Label7.Caption:='0';

Label8.Caption:='0';

Label9.Caption:='0';

Label10.Caption:='1';

Label12.Caption:='8'; end;

If Edit19.Text=('80') Then Begin

Led1.Brush.color:=ClRed;

Led2.Brush.color:=ClWhite;

Led3.Brush.color:=ClWhite;

Led4.Brush.color:=ClRed; Label7.Caption:='1'; Label8.Caption:='0'; Label9.Caption:='0'; Label10.Caption:='1'; Label12.Caption:='9'; end; If Edit19.Text=('96') Then Begin Led1.Brush.color:=ClWhite; Led2.Brush.color:=ClRed; Led3.Brush.color:=ClWhite; Led4.Brush.color:=ClRed; Label7.Caption:='0'; Label8.Caption:='1'; Label9.Caption:='0'; Label10.Caption:='1'; Label12.Caption:='10'; end; If Edit19.Text=('112') Then Begin Led1.Brush.color:=ClRed; Led2.Brush.color:=ClRed; Led3.Brush.color:=ClWhite; Led4.Brush.color:=ClRed; Label7.Caption:='1'; Label8.Caption:='1'; Label9.Caption:='0'; Label10.Caption:='1'; Label12.Caption:='11'; end;

If Edit19.Text=('192') Then Begin

Led1.Brush.color:=ClWhite; Led2.Brush.color:=ClWhite; Led3.Brush.color:=ClRed; Led4.Brush.color:=ClRed; Label7.Caption:='0'; Label8.Caption:='0'; Label9.Caption:='1'; Label10.Caption:='1'; Label12.Caption:='12'; end; If Edit19.Text=('208') Then Begin Led1.Brush.color:=ClRed; Led2.Brush.color:=ClWhite; Led3.Brush.color:=ClRed; Led4.Brush.color:=ClRed; Label7.Caption:='1'; Label8.Caption:='0'; Label9.Caption:='1'; Label10.Caption:='1'; Label12.Caption:='13'; end; If Edit19.Text=('224') Then Begin Led1.Brush.color:=ClWhite; Led2.Brush.color:=ClRed; Led3.Brush.color:=ClRed; Led4.Brush.color:=ClRed;

Label7.Caption:='0';

NIVERSITY LIBRARY 1988

Label8.Caption:='1'; Label9.Caption:='1'; Label10.Caption:='1'; Label12.Caption:='14'; end; If Edit19.Text=('240') Then Begin Led1.Brush.color:=ClRed; Led2.Brush.color:=ClRed; Led3.Brush.color:=ClRed; Led4.Brush.color:=ClRed; Label7.Caption:='1'; Label8.Caption:='1'; Label9.Caption:='1'; Label10.Caption:='1'; Label12.Caption:='15'; end; If Edit19.Text=('0') Then Label13.Visible:=True else Label13.Visible:=False; IF sonuc>0 Then Begin Label13.Visible:=False; Label14.Visible:=True; end else Label14.Visible:=False; If sonuc=240 Then Begin Label15.Visible:=True; end

Else

Label15.Visible:=False;

end;

procedure TForm1.Button22MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,2 mov dx,378h out dx,al end;

procedure TForm1.Button22MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end;

procedure TForm1.Button23MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm

```
mov al, 1
mov dx,378h
out dx,al
end;
end;
```

procedure TForm1.Button24MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,4 mov dx,378h out dx,al end;

procedure TForm1.Button25MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,8 mov dx,378h out dx,al end;

procedure TForm1.Button23MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end; end;

procedure TForm1.Button24MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end; end;

procedure TForm1.Button25MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin asm mov al,0 mov dx,378h out dx,al end;

end;

procedure TForm1.SpeedButton1Click(Sender: TObject);

begin Memol.Lines.Clear; end;

procedure TForm1.GroupBox2Click(Sender: TObject); begin

end;

end.

5. PLC PROGRAMS

5.1. Introduction

Siemens S7 200 212 CPU has 8 inputs $(I_{0.0}, I_{0.1}, \dots, I_{0.7})$ and 6 output $(Q_{0.0}, Q_{0.1}, \dots, Q_{0.5})$. Since these output and inputs were insufficient for our purpose I made the modifications as it can be seen in the figures 1.7a and b and turned the system into a single PLC.



Fig.1.7aPLC's



Fig.1.7b

5.2. Ladder Diagram

Each network of the Ladder program is individually explained. ladder diagram is shown.

c:\microwin\projects\gproje1.ob1

PROGRAM TITLE COMMENTS

Press F1 for help and example program

Network 1 Control İnput For The Heater

Counter counts from Co to maximum 11. Controls are made with two up and down buttons. I 0.6 and I 0.7 heater counters are adjusted to count up to ten by equalizing PV+1. The reset button is used in parallel with the heater count reset to allow the system to reset after 10.



64
Network 2 Multiplication

Each value entered into counter is multiplied by 100. these are registered at AC1. This function is repeated each time by the I0.5 sensor. For each packaging the figures are entered to the multiplier at AC1. the values are multiplied by 100. the maximum figure that can be registers is 10. this is equal to 10 seconds



Setwork 3 Air Cleaner

There is a fan for cleaning the air of the packaging unit. The fan is switched on by a delta starter switch. This is the first unit that is run when the system starts to operate. All the security buttons of the whole system has a parallel connection to the IOO stop button. T40 and a sensor avoids the system to work when it is not necessary. Since there is no other output possibility star and delta are jointly used as the lock for the system. The passage time from star to delta is 10s.



Network 4 First Pallet

When the system is initiated the star and delta outputs energize the unit 4. for the first pallet to work fan motor has to be used as delta. The buttons on the first and second silos should then be switched to on position. When the first silo is full up the sensor on the second silo is automatically turned to on. When the switches are on the pallet starts to work.



Network 6 Security Of The Nylon Roll

There is a danger that if the IO.5 sensor does not see the black marks the whole role of nylon can be run off and wasted. To avoid such a situation there is an emergency system that when the motor runs more than ten seconds non-stop, this means that there is something wrong and the system is automatically switched off.



Network 7 Heater

When the I0.5 sees that the nylon is at the required place the welding bar activates and welds the nylon at the required place.





55 69



56 70

c:\microwin\projects\gproje1.ob1 Network 12 End Of Ladder Program (END) 57 71

TITLE COMMENTS

for help and example program

1 Input For Counter

is for count up and IO.1 is for count down. When the both buttons are pressed



5 of the first PLC counts the process in order to obtain as many output as cated at the C48 counter. A pulse is sent for each stage. And when I0.5 and starts CO becomes reset.

a server a server a server a



*

Timer

the system reaches the registered number of operations it needs to be switched a timer is used to stop the system after a certain period.



desired number of packaging is made 24VDC is sent to the I0.0 output of the C. This increases the contact

ما بعد المالي المالي المالي المالي المالية المالية المالية المالية المالية المالية المالية المالية ال



5 Emergency

e of emergency 30ms direct current is given to stop all the motors. PLC 1 0.0 to off position by giving current to the Q0.0



required number of packaging is made the whole system stops.



ork 7 For Output



Q0.5

*

9 End Of Ladder Program

.62 76

5.3. Statement List

First PLC Program Title Comments

NETWORK 1 //Control input for the heater

//Counter counts from Co to maximum 11. Controls are made with two up and down buttons. I O.6 and I 0.7 heater counters are adjusted to count up to ten by equalizing PV+1. The reset button is used in parallel with the heater count reset to allow the system to reset after 10.

LDI "stop_switch" LD "heater_counter_down" O C0 CTU C0, +11

NETWORK 2 //Multiplication

//Each value entered into counter is multiplied by 100. these are registered at AC1. This function is repeated each time by the I0.5 sensor. For each packaging the figures are entered to the multiplier at AC1. the values are multiplied by 100. the maximum figure that can be registers is 10. this is equal to 10 seconds

LD "limit_sensor" MOVW C0, AC1 MUL +100, AC1 TON T33, AC1

NETWORK 3 //air cleaner

//There is a fan for cleaning the air of the packaging unit. The fan is switched on by a delta starter switch. This is the first unit that is run when the system starts to operate. All the security buttons of the whole system has a parallel connection to the I00 stop button. T40 and a sensor avoids the system to work when it is not necessary. Since there is no other

output possibility star and delta are jointly used as the lock for the system. The passage time from star to delta is 10s.

LDN "stop switch" AN T40 LD "start switch" 0 "star output" 0 "delta autput" ALD LPS A "star output" AN "delta_autput" TON T37, +100 LRD AN "delta_autput" AN T37 -"star output" LPP LD T37 0 "delta_autput" ALD AN "star output" Probane Probane "delta autput"

NETWORK 4 //First Pallet

//When the system is initiated the star and delta outputs energize the unit 4. for the first pallet to work fan motor has to be used as delta. The buttons on the first and second silos should then be switched to on position. When the first silo is full up the sensor on the second silo is automatically turned to on. When the switches are on the pallet starts to work.

- LD "delta_autput"
- A "level_sensor1"

= "motor_for_1.palet"

NETWORK 5 //Nylon bag wrapper

//When the Q0.2 is in action and the two silos are full, I0.5 sensors are opened to register black signs. These then start the T39when the sensor does not see the band I0.5 becomes on. When the band rotates and the black sign is apparent the I0.5 Q0.4 output is cut off.

LD "delta_autput"

LDN "limit sensor"

O T39

ALD

A "level sensor2"

= "motor_for_naylon_roll"

NETWORK 6 //Security of the Nylon roll

//There is a danger that if the I0.5 sensor does not see the black marks the whole role of nylon can be run off and wasted. To avoid such a situation there is an emergency system that when the motor runs more than ten seconds non-stop, this means that there is something wrong and the system is automatically switched off. Network 7://Heater

LD "motor_for_naylon_roll"

O T40

A "delta_autput"

TON T40, +200

NETWORK 7 //Heater

//When the I0.5 sees that the nylon is at the required place the welding bar activates and welds the nylon at the required place.

LD "delta_autput"

A "limit_sensor"

AN T33

= "heater_output"

NETWORK 8 //Filling

//The discharge notor Q0.0 is not active during the welding of nylon takes place. The Q0.0 motor becomes active when I0.3 limit switch is on and the silo 2 is full.

LD "delta_autput"

LD "limit_switch"

ON T38

ALD

- AN "motor_for_naylon_roll"
- AN "heater_output"
- A "level_sensor2"
- = "main_output"

NETWORK 9 //Timing

//I0.3 switch decides when the Q0.0 motor should stop. When the system is about to empty I0.3 is in off position. In order to activate the Q0.0 motor the I0.4 limit switch must be on. T38 30 ms gives the initial movement.

LD T33

AN "heater_output"

TON T38, +30

NETWORK 10 //Revolution timing

//After the discharge took place counter enables Q0.4 to place a new bag in the position. After the first revolution T39 counter starts counting. Safety mechanism avoids the system to continue when the I0.4 is not properly functioning. LDN "limit_switch" LD "heater_output" CTU C1,+1

NETWORK 11 //Timer

//Timer initiate the Q0.4 movement abd provides safety to Q0.0.

LD C1 TON T39,+30

NETWORK 12 //End of lader program

MEND

Second PLC Program Title Comments

NETWORK 1 //Entry o pallet number

//I00 is for count up and I0.1 is for count down. When the both buttons are pressed simultaneously the counter is reset.

LD "counter_up"

LD "counter_down"

LD "counter_up"

A "counter_down"

CTUD C48, +999

NETWORK 2 //Limit

//Q0.5 of the first PLC counts the process in order to obtain as many output as indicated at the C48 counter. A pulse is sent for each stage. And when I0.5 and I0.0 starts CO becomes reset.

LD "count_up_sensor" LD "counter_down" A "counter_up" O C0 CTU C0, C48

NETWORK 3 //Timer

//When the system reaches the registered number of operations it needs to be switched off. A timer is used to stop the system after a certain period.

LD C0 O "output_for_off" TON T39, +20

NETWORK 4 //Output of the Counter

//When the desired number of packaging is made 24VDC is sent to the I0.0 output of the first PLC. This increases the contact

LDN T39 LD C0 O "output_for_off" ALD = "output_for_off"

NETWORK 5 //Emergency

//In case of emergency 30ms direct current is given to stop all the motors. PLC 1 turns I0.0 to off position by giving current to the Q0.0

"emergency" LD O C49 TON T38, +30 to a second the externation of the restoucts to these are not AN T38 = "emergency_output"

NETWORK 6 //Service counter //When the required number of packaging is made the whole system stops.

LD "count_up_sensor" LD "count_down_service" LD "service_reset" CTUD C49, +1000

NETWORK 7 // For output

LD "count_up_sensor" = Q0.4

NETWORK 8 // For output

LD 10.6

= Q0.5

.

//End of ladder program **NETWORK 9**

MEND

CONCLUSION

The aim of the study is showing that any process can be managed remotely with ease. Need for remote managing could appear in health-critical or dangerous conditions, being far away from job, etc. It could be extremely useful for managers to check or administer or just for taking information as if using visual phone.

In fact, a production unit may have a PLC or other types of control device on their processes, so that they may not need this part of job. In this case, if some computers make the data collection and control the process with modem/internet availability, then server and client side programs are applicable for them.

In our study, since the work done in local area network, the process control by client realized in less than 1 second, but of course, in a wide area network connection would take more time due to distance and the quality of computers and their peripherals. Our study worked from modem to telephone line successfully. But operation occurred slowly. From local area network, more fast than wide area network.

Our study is possible to send mobile telephone message or e-mail but we must add something to the software. It is reality that my project supply extra security to my study area. We can increase security to use technology and we can control all system whatever we went from the away.

I believe that the proposed work is an economic and useful solution for large and distributed companies, and workshops, which have to work together, even for institutions having health critical or dangerous jobs.

REFERENCES

Lange, L., 1998. Analysis & Forecast: The Internet. IEEE Spectrum, 35(1): 37-42.

Dutta-Roy, A., 1999. Networks for Homes. IEEE Spectrum, 36(12): 26-33.

Dutta-Roy, A., 1999. Bringing Home the Internet. IEEE Spectrum, 36(3): 32-38.

Yourdon, 1997. Analysis & Forecast: Software Engineering Viewpoint, IEEE Spectrum, 34 (1):66-67

Choi, S.Y., and Whinston, A.B., 1999. The Future of E-Commerce: Integrate and Customize. IEEE Computer, 32(1):133-138.

Floyd. Electronic Devices

TAŞBAŞI Abdurrahman PC Donanımı

THOMAS L. Floyd. Digital Fundamentals Sixth Edition

THOMAS L. Floyd. Electronic Devices Fourth Edition

ALTINBAŞAK Orhan . Bilgisayar Uygulamaları

Electronics Workbench

Boardmaker

Rsview32: http://www.software.rockwell.com/rsview32.

Cimplicity: http://www.gefanuc.com/cimplicity/index.shtml.

Factory suit webserver: http://www.wonderware.com/products/fswebsrvr71.htm.

Wince Siemens: http://www.wincc.de.

PC-Anywhere: http://www.symantec.com/pcanywhere/index.html