NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

PROGRAMMABLE LOGIC CONTROLLERS (PLC)

Graduation Project
EE 400

Student: A. Bahadır Soysal (940869)

Supervisor: Mr. Özgür Özerdem

T.R.N.C. - 2000

CONTENTS

ACKNOWLEDGEMENTS

I must first acknowledge some depts from my more distance past. To my parents for their support to an upwardly mobile son. To my friend Erkan Hakverdi, Yudum Ceylan, Şaban Dağ for his helps.

I want to thanks separately to my teachers that during my education they support me, the Dean of Engineering Faculty Prof. Dr. Halil ISMAILOW, Head of Electrical & Electronic Department Prof. Dr. Fakhraddin MAMEDOW and also Mr. Özgür ÖZERDEM, Mr. Tayseer ALSHANABLEH and Mr. Kaan UYAR.

And finally, I want to thanks everybody who helped and supported me to come to todays and graduated from a university.

ABSTRACT

PLC (Programmable Logic Controllers) is a thing that programmable with computer support to take more efficiency from time and workers. It is divided into two parts. Hardware and software.

The hardware are the parts of machine those are CPU, I/O device and Programming device. CPU is basic microprocessor system and it carries out as control sensor, counter, timer function. CPU carries out stored user program in memory will input informations from various sensor circuits and can sending suitable output to commands and control circuits. I/O Module receives 120 VAC signal in device or processing device and transforms 5 VDC signal form.

There are many specialisation such as timer, counter, master control set, which works data and controls program, master control reset, JMP. There are command which are mathematics process that are comparator processes. These are the main function and feature of software part of PLC.

INTRODUCTION

In the late 1960's PLCs were first introduced. The primary reason for designing such a device was eliminating the large cost involved in replacing the complicated relay based machine control systems. Bedford Associates (Bedford, MA) proposed something called a Modular Digital Controller (MODICON) to a major US car manufacturer. Other companies at the time proposed computer based schemes, one of which was based upon the PDP-8. The MODICON 084 brought the world's first PLC into commercial production.

When production requirements changed so did the control system. This becomes very expensive when the change is frequent. Since relays are mechanical devices they also have a limited lifetime which required strict adhesion to maintenance schedules. Troubleshooting was also quite tedious when so many relays are involved. Now picture a machine control panel that included many, possibly hundreds or thousands, of individual relays. The size could be mind boggling. How about the complicated initial wiring of so many individual devices! These relays would be individually wired together in a manner that would yield the desired outcome.

These "new controllers" also had to be easily programmed by maintenance and plant engineers. The lifetime had to be long and programming changes easily performed. They also had to survive the harsh industrial environment. That's a lot to ask! The answers were to use a programming technique most people were already familiar with and replace mechanical parts with solid-state ones.

In the mid70's the dominant PLC technologies were sequencer state-machines and the bit-slice based CPU. The AMD 2901 and 2903 were quite popular in Modicon and A-B PLCs. Conventional microprocessors lacked the power to quickly solve PLC logic in all but the smallest PLCs. As conventional microprocessors evolved, larger and larger PLCs were being based upon them. However, even today some are still based upon the 2903. Modicon has yet to build a faster PLC than there 984A/B/X, which was based upon the 2901.

Communications abilities began to appear in approximately 1973. The first such system was Modicon's Modbus. The PLC could now talk to other PLCs and they could be far away from the actual machine they were controlling. They could also now be used to send and receive varying voltages to allow them to enter the analogue world. Unfortunately, the lack of standardisation coupled with continually changing technology has made PLC communications a nightmare of incompatible protocols and physical networks. Still, it was a great decade for the PLC!

The 80's saw an attempt to standardise communications with General Motor's manufacturing automation protocol (MAP). It was also a time for reducing the size of the PLC and making them software programmable through symbolic programming on personal computers instead of dedicated programming terminals or handheld programmers. Today the world's smallest PLC is about the size of a single control relay!

The 90's have seen a gradual reduction in the introduction of new protocols, and the modernisation of the physical layers of some of the more popular protocols that survived the 1980's. The latest standard (IEC 1131-3) has tried to merge plc-programming languages under one international standard. We now have PLCs that are programmable in function block diagrams, instruction lists, C and structured text all at the same time! PC's are also being used to replace PLCs in some applications. The original company who commissioned the MODICON 084 has actually switched to a PC based control system.

SECTION I

1.1. THE TYPES OF PLC

In general, PLC divides to three sections;

*Central Processing unit(CPU)

*The input/output section

*The programming device

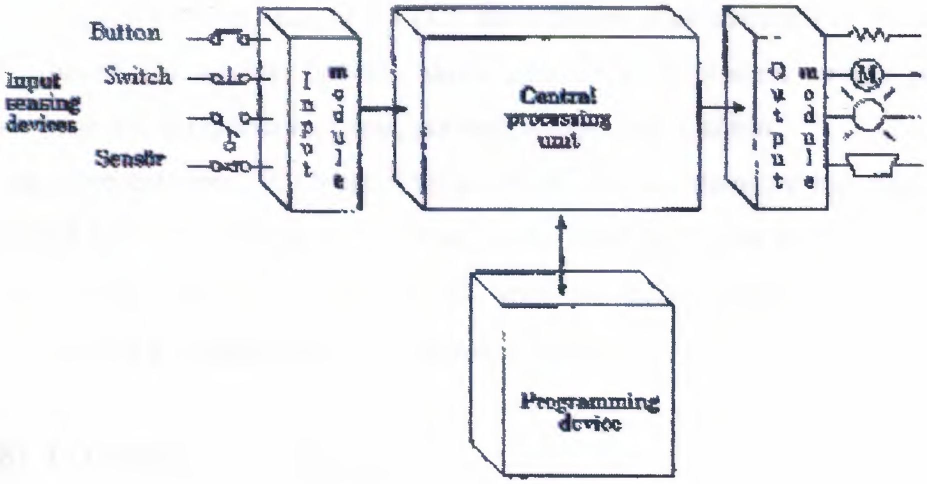


Figure.1.1.1. PLC sections

(CPU), PLC system and there are various logic circuit gates. CPU is basic microprocessor system and it carries out as control relay, counter, timer functions. CPU carries out user programs stored in memory and read input data from various sensor circuits and can send suitable outputs to commands and to control circuits.

Direct current power supply must be used for the low level voltage that these are using in processor and I/O models. This power supply is a part of CPU. PLC system is independent in its structure and also it can be dependent to its system.

I/O system forms can be connected to controller by other devices. The aim of interface is to send various signals and to take situations to external devices. The output devices for example, motor starters, solenoid valves, indicator lights connected to terminals on the output module.

The desired program loads to processor's memory by programming device or terminal. This program can enter to relay during using ladder logic. Program can be obtained till the main control or machines by sequential processes.

a) PLC size and practice:

There are 3 different categories of PLC; as small, medium and large.

*In small group category, PLC has bigger than input/output of 128 I/O and bigger than memory of 2 KB.

*In medium group category, PLCs have bigger than memory of 32 KB and 2048 I/O. Special I/O module provide easily adaptation in process control practice, analog functions like temperature, press, current, weight and position.

*In large category, PLCs have bigger than 750 KB memory and bigger than input/output of 8192 I/O. This group is for unlimited practice to give force.

Nowadays, PLCs are used in all area of industry along in chemistry, automotive industry production of steel and paper factory.

b) I/O unit:

I/O unit forms is the input/output rack of PLC. I/O unit receives 120 Vac signal in device or processing devices and transforms 5 Vdc signal form. In output units controller signals (5Vdc) are used to devices or processor control as 120 Vac. These output signals provide low current control that used in power electronic elements or optic isolators. Input/output unit in PLC can be put in the same structure or different structure with CPU. This standard input/output unit is in the following shape.

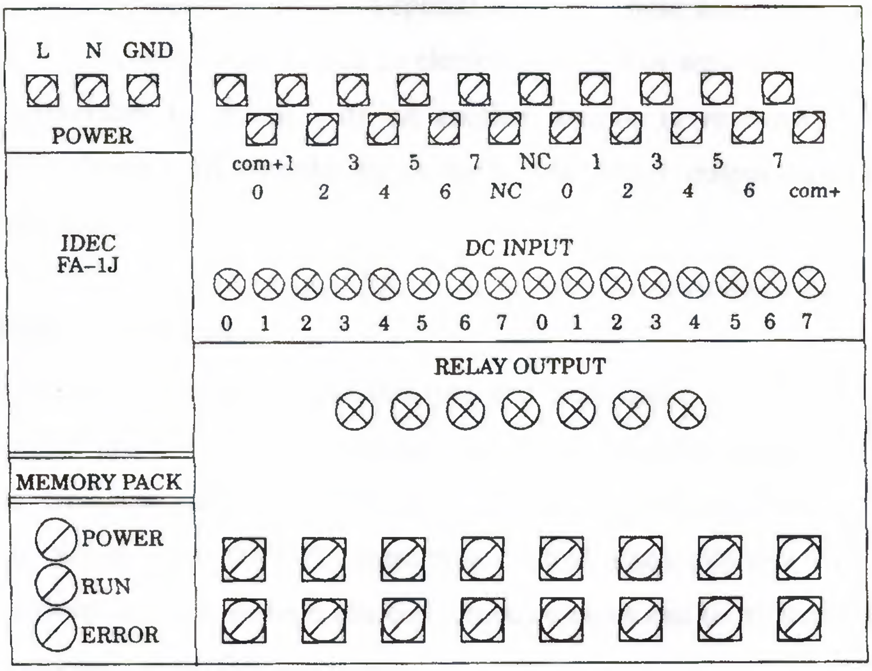


Figure. 1.1.2. In the same structure CPU with PLC I/O unit

Between processor and I/O rack communication different connection cables are permitted. This condition is as the following figure 1.1.3.

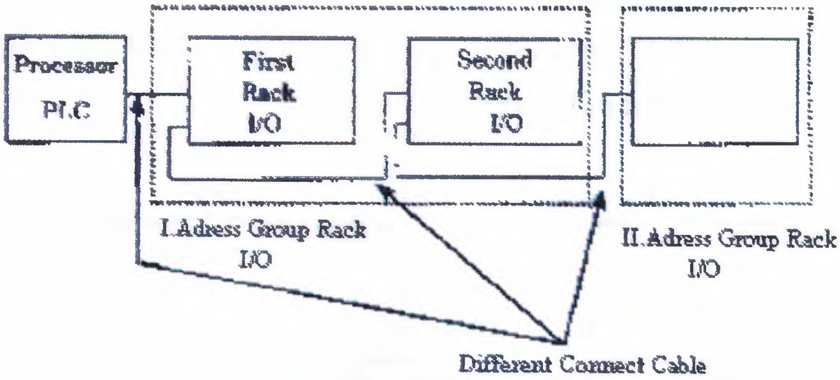


Figure.1.1.3 Between Processing I/O Racks communication

I/O units each input/output has a special address. These addresses are known by the processor. To connect output/input an element with I/O or separating is very easy and quick. Furthermore to change with an another module is very easy. ON/OFF condition of I/O circuit each module shows with light. Many output modules have rubbish fuse indicator.

c) Different I/O units:

Many output I/O units are from this type and most useful is interface module. This type interface provides to link of inputs as selector switches push buttons and limits switches. However, output control lights small motor solenoids sensor and motor starters limit it. Which have ON/OFF contacting control. Each different I/O module takes its power from common voltage sources. These voltages can be different size and type. These are showed in the following table.

Input Interface	Output Interface
24Vac/dc	12-48Vac
48Vac/dc	120Vac
120Vac/dc	230Vac
230Vac/dc	120Vdc
5Vdc (TTL level)	230Vdc
	5Vdc (TTL level)

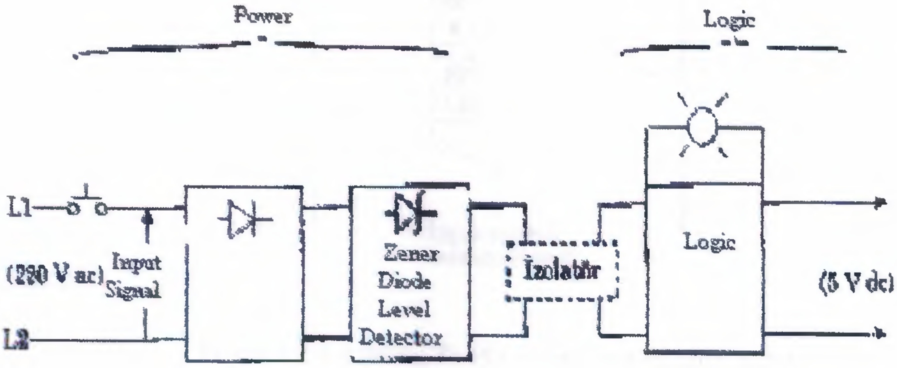


Figure.1.1.4 AC input interface block diagram

Shows that entries block diagram for an alternative current to input module. Input circuit compose of to main section as power and logic section.

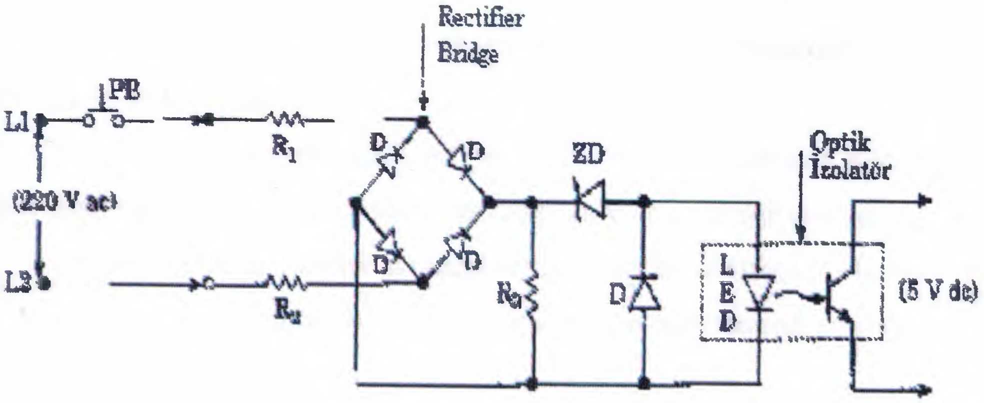


Figure.1.1.5. Simplified Circuit For a AC Module

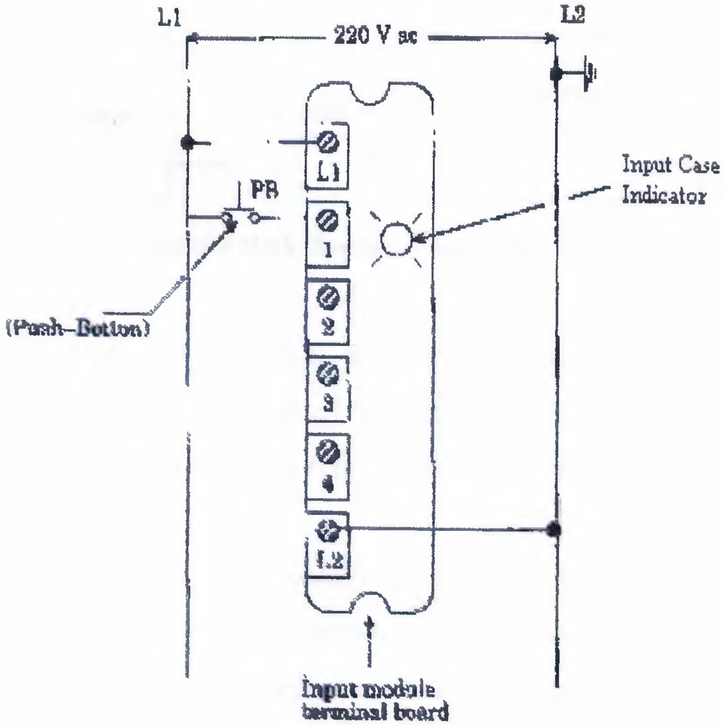


Figure.1.1.6. Linking To PLC Input Unit of 220V Input

Figure 1.1.4 and 1.1.5 shows figural diagram of Ac input module for input, also figure 1.1.6 shows connect terminal.

When push button shuts down, bridge type treatment exercise 220V AC voltage from R_1 and R_2 resistance's.

Zener diode (ZD) voltage limit regulates according to low level voltage.

When light come to processor from led with phototransistor that means low level voltage (SV'dc) is transmitted.

Optic isolator separates high AC voltage from logic circuits also protects to processor from damages, which comes from temporary line voltage change.

Furthermore, optic isolator protects to processor from effect of electrical noise.

Kuplaj and isolation can be created with using a pulse transformation.

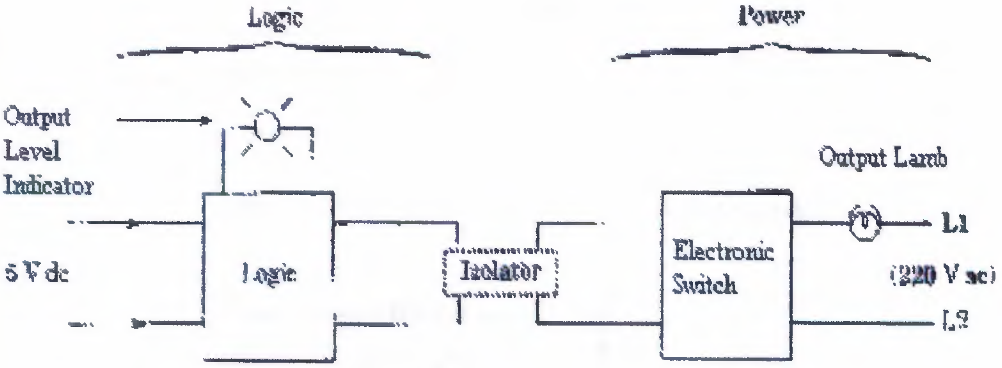


Figure. 1.1.7 typical a block diagram of output interface module.

Figure. 1.1.7 shows typical a block diagram of output interface module. Also output module, as input module, composes of two departments such as power and logic.

Device in output is controlled by the 5V comes from logic unit. In this unit, processor sets output conditions.

When processor, led, in optic isolator, distributing light exercises an output voltage (5V' dc), however, phototransistor is switching and conducting. This means that to detect and conduct of triac, and lamp, that uses as output element, turn on ON condition.

When led in logic unit turn off, logic become 0 condition and phototransistor cannot conduct. If a DC device in output will be controlled, it is carried with circuit.

PLC device will not be damaged from optic isolation that will be from power department.

If many high fast ON-OFF is necessary, in right current transistor and also alternative current triac circuits are used. Current cannot pull on PLC from output modules. Maximum current capacity of each device exists in their catalogs of that model.

In high currents instead of triac or other effect elements, standard relay must use as table 6. There are output/input unit as analog/digital translator (ADC) and digital/analog translator (DAC) that it is necessary for feedback control exercises in PLC devices.

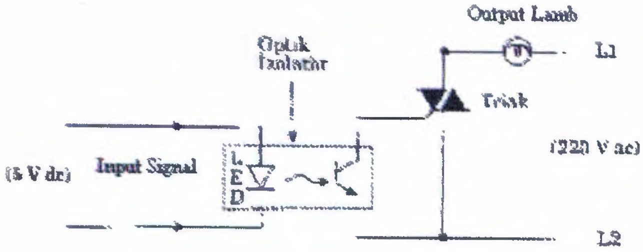


Figure. 1.1.8 Simplified circuit of an AC output module.

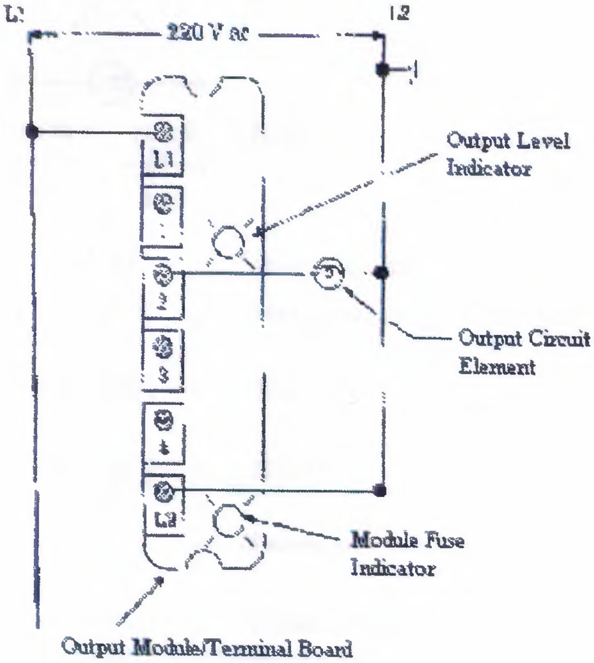


Figure. 1.1.9 Internal wire connection typical an output module

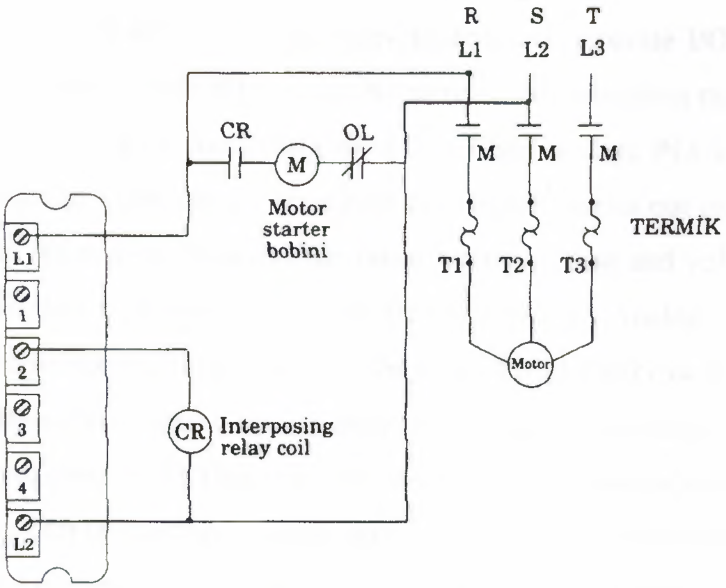


Figure. 1.1.10 Sensor connection points

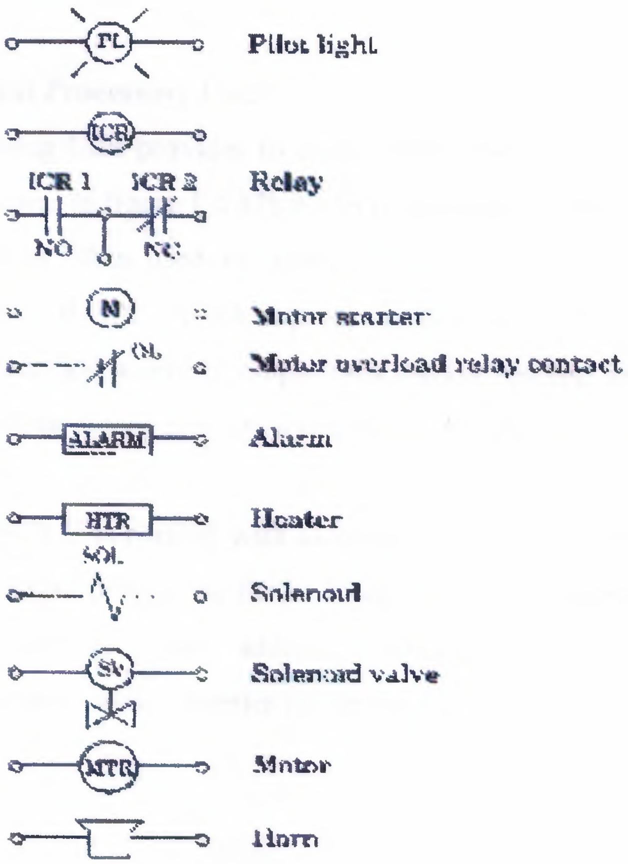


Figure. 1.1.11 Symbols of output control circuit

d) Analog input/output unit (I/O modules):

First produced PLCs only had been limited with separate I/O interfaces which had been allow to link to ON/OFF device. Because of this limitation many of processing exercises could be as part controlling by PLC. Also in days PLCs included analog interface and separate (I/O) input/output interface, which carries out practically many of control process. An analog input module takes analog current and voltage that is taken off analog input and it changed to digital data form by an Analog Digital Converter (ADC). In this condition turning levels are shown as 12-bit binary or 3 digit BCD that is rates with analog signal. Analog sensor elements are transducers as heat, light, velocity, pressure, and wet sensors. All these sensors can be linked to analog input

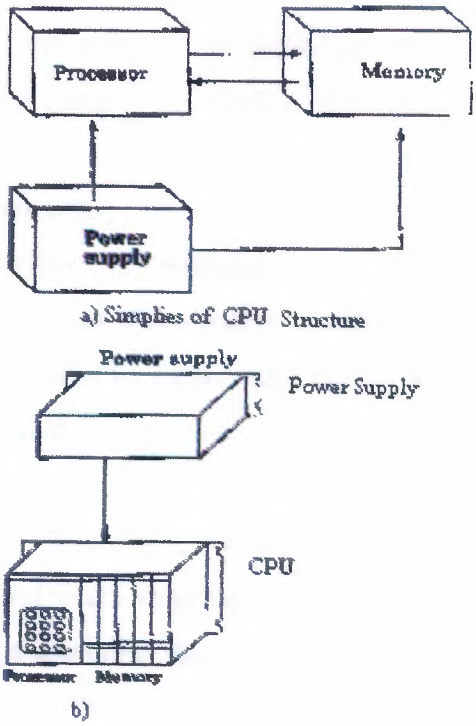
Analog output interface module takes digital data from processor, charges rate with voltage and current and controls a device as analog. As a whole digital data passes from Digital/Analog output device are small motors, valves and analog measure devices.

e) CPU (Central Processing Unit):

Central Processing Unit provides to communicate between power supply and processor memory modules. In figure 1.2.12b it can find covered both of two units.

CPU statement is often used as mean of processor statement. Processor-memory creates a big unit of CPU, which is programmable brain of controller. In this unit, there are microprocessor, memory chips, information reading and request data from memory, programming device and communication circuits, which is necessary for processor.

Development of PLC is parallel with increasing especially of CPU. In our day PLC systems carry out logic processing furthermore they have some especially such timer, counter, data storing, main addition-subtraction, multiplication-division processes, compare processes, code converter processes.



a) Simplies of CPU Structure

b)

Figure. 1.1.12 CPU; the elements of central processing unit (a) the structure of simplified CPU (b) power supply unit different from CPU.

f) Processor-Memory Module:

CPU is the brain of programmable of controller and a big part of CPU family forms from processor memory unit. This module cover microprocessor, memory chips programming device and necessarily communication circuits for processor interface.

Furthermore processor carries out other functions. For example, it carries out timer, counter, compare, keeper and addition, subtraction, multiplication and division functions, which are four main functions of mathematics.

1.2. MEMORY DESIGN

Memory is used to store data. This stored information is related with which output sign will be store as, which shows input, and the structure of program necessary amount of memory. It stores special information parts, which is named as memory bit. 1 byte = 8 bit, 1024byte = 1kbyte and the number of memory capacity is stated these units.

The memory types are divided into two groups;

The first group: the energy of power supply is cut that supplied memory, it means that memory had been erased. Also second group: hide information cannot lose if the energy is cut. But to change of includes of those types of memories, there is a necessary a special system.

a) I. Group Memories:

First group memories are Random Access Memory (RAM) and Read/Write (RIW). In these types memories if the energy is cut, the information is lost. If RAM is supplied program can be stored by battery that battery is in PLC device. When battery energy finishes, program will be erased.

b) II. Group Memories:

It is Read Only Memory (ROM). The type memory can be erased and programmable. It is divided four into groups;

1) PROM (Programmable Read-Only Memory): it is a special type of ROM. PROM memory allows to writing of information in chip, these information are provided or there were at the beginning. The information can be written into ROM only one time.

The main disadvantage of PROM is no erasable and no Programmable. In PROM programming is doing as dissolve and pluck logic, for this reason, the erasing of erasable connections is process that there is no to turn back. For this reason, firstly all mistake control process must be finished.

2) EPROM (Erasable Programmable Read-Only Memory): this type is the memory type that is used in PLC devices. Written programmable firstly, is store in EPROM memory and is sent central processing unit.

3) EAROM (Electrically Alterable Read-Only Memory): It is like EPROM memory, but to erase and ultraviolet light supply is not necessary. EAROM chip to clean by erasing, an eraser voltage is exercised to suitable pin. When chip erases one time, it can be programmed again.

4) EEPROM (Electrically Erasable Programmable Read-Only Memory): In EEPROM memory type, when energy is cut, information cannot lose as EPROM. Special device is not necessary in writing and erasing processing. EEPROM or EPROM memories that are mounted to PLC make runs as stored program into records.

Data table stores information's, that are necessary to carry to the program, which includes information's such as output and input conditions, timers, and counter results and data records. Includes of table is divided two groups as conditions data and numbers (or codes) 0 and 1 conditions are ON/OFF conditions of information that records the place of bit. Data table is divided 3 sections. Input view table stores the condition of digital input that relations input interface circuits. As ON/OFF condition, in this unit results of input are stored as zero (0) or one (1).

Output view memory is order of bits that control the digital condition of devices which links interface of output. The logic conditions of output units are stored in this memory and it is taken from this logic level memory and transfers to output unit.

1.3. PROGRAMMING DEVICES

The most important one of features of programmable controller is to have programming elements, which are useful. Programming device provides transformation between operator and circuit of controller. (Fig. 1.3.1)

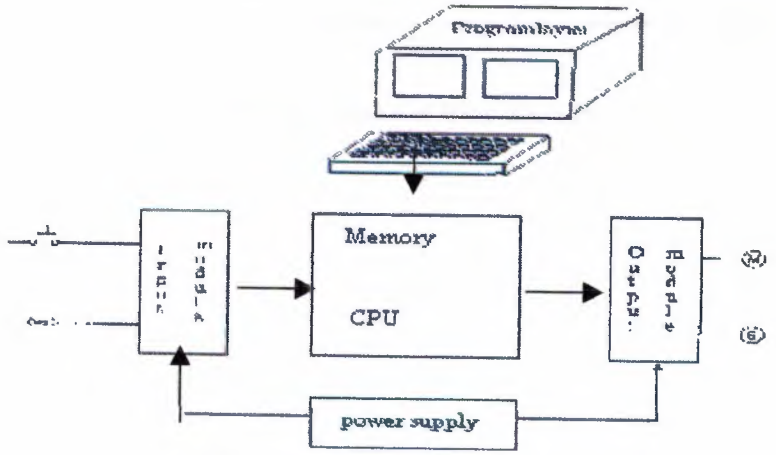


Figure. 1.3.1. Transformation of PLC Circuits

Programming terminal relation between PLC memory and monitor. User sends programming device and PLC control program to device.

Generally, industrial CRT terminals in many devices are used for programmable controllers. These terminals include indicator units, keyboards and CPU and they provide to communicate necessary order.

The advantage of CRT is to check program is easily on monitor.

In small PLCs programming is used cheap, moveable, small and mini programmable devices. The monitor of this type of programming monitor is liquid crystal screen instead of CRT tube, which name LCD. On mini program there are LCD monitor program coding keys and special functions keys. FA2 of programming device IDEC FA1 Junior module is shown at table 1.3.2.

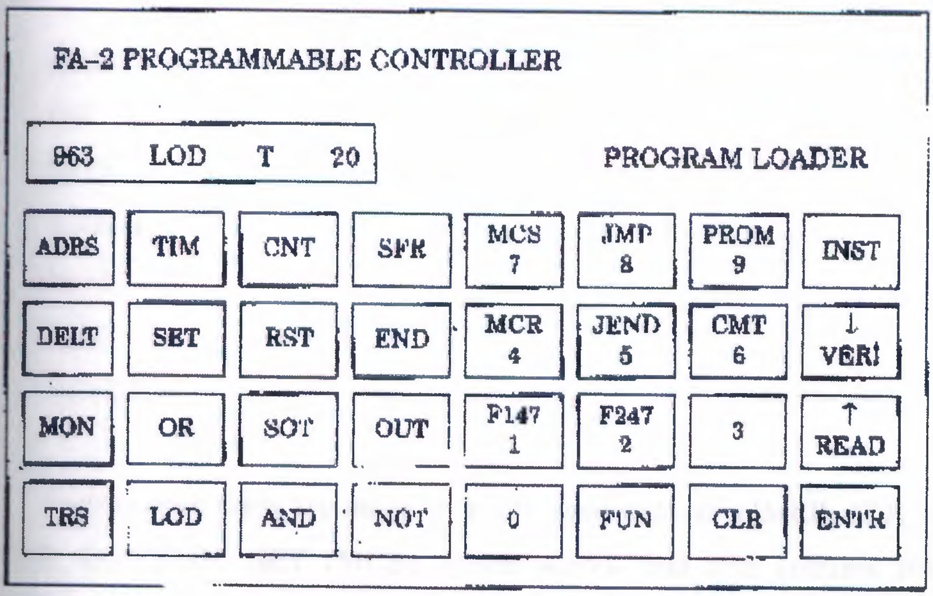


Figure. 1.3.2. Programming Device of IDEC FA-1 PLC.

SECTION II

2.1. PLC PROGRAMMING SOFTWARE

In this section, PLC programming fundamental is prepared, student's capacity, which met PLC programming, is considered first time.

AND

OR

NOT

NAND

NOR

SET

RESET

Furthermore there are many specialisations such as TIMER, COUNTER, and MASTER CONTROL SET (MCS), which works data and controls PROGRAM, MASTER CONTROL RESET (MCR), JMP. There command which are mathematics process that are comparator processes (=, <, >).

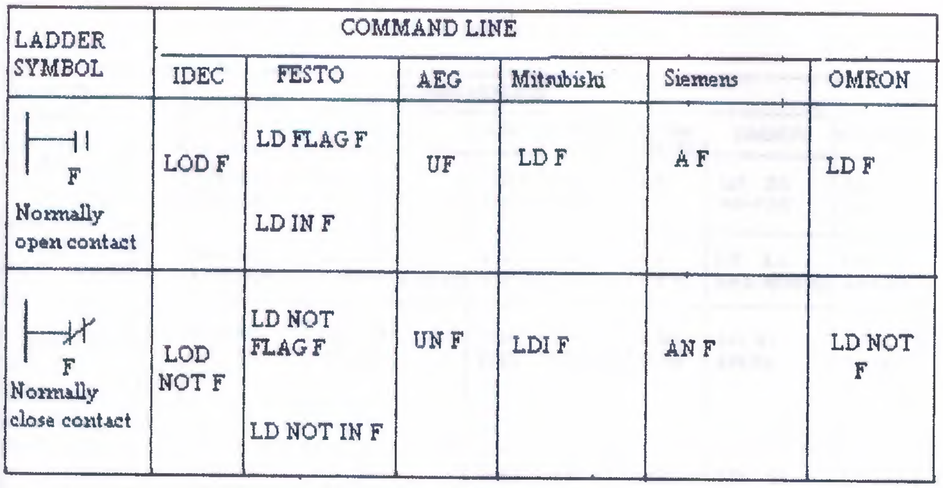
In all PLC systems, to create logic process is programmed as the same are carried out some function. However, the main logic is the same that TIMER, COUNTER and SHIFT REGISTER functions are to get command and programmed but there can be some differences.

2.2. CREATE OF LEADER DIAGRAM

a) Start Commands:

These commands are first element of program. There are two type contact conditions as at table 2.2.1. First normally is open also second close.

Normally, starting with open contact this program command is to get command as LD IN, LD, LOD A, on PLC device. And also close contact is stated as LDI, LD NOT, LOD NOT, AN.



LADDER SYMBOL	COMMAND LINE					
	IDEC	FESTO	AEG	Mitsubishi	Siemens	OMRON
F Normally open contact	LOD F	LD FLAG F LD IN F	UF	LD F	A F	LD F
F Normally close contact	LOD NOT F	LD NOT FLAG F LD NOT IN F	UN F	LDI F	AN F	LD NOT F

Table 2.2.1. Load Exercising

Note: in table F value is constant and input/output interval relay, special relay, timer, counter can be SFR number.

According to this table at MITSUBISHI and HITACHI model normally open contact is shown with LD, also close contact is shown with LDI.

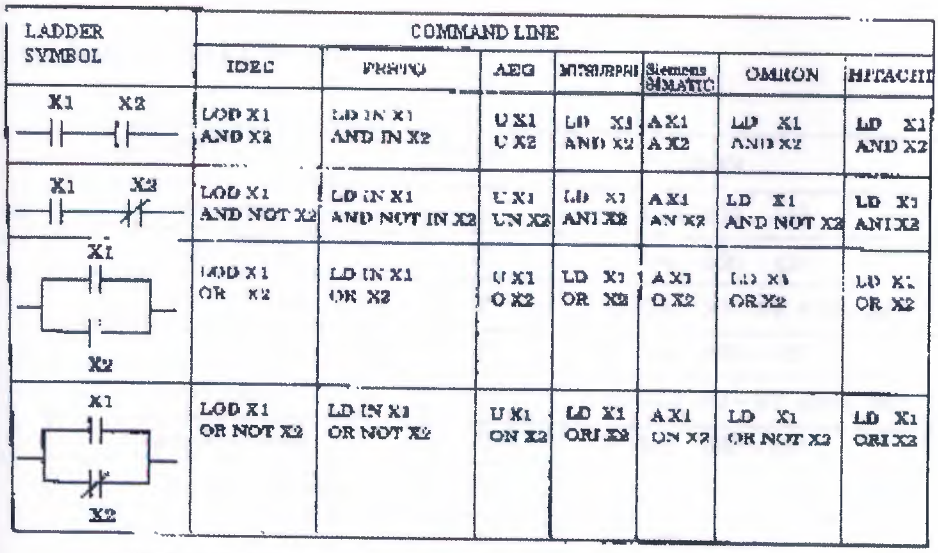
Also at AEG PLC, U (UND) command is used for open contact and (UN) UND-NICHT command is used for closed contact.

Also at SIEMENS PLC, A (AND) command is used to open contact and AN (AND-NOT) is used for closed contact.

At OMRON PLC, open contact is shown LD, also close contact is shown with LD NOT.

Also at FESTO PLC, open contact LD FLAG is used for flag load other conditions LD IN command is used to contact load. In normally, also close contact is programmed for flag exercising as LD NOT FLAG... For other contacts are programmed as LD NOT IN...

b) AND and OR Exercising:



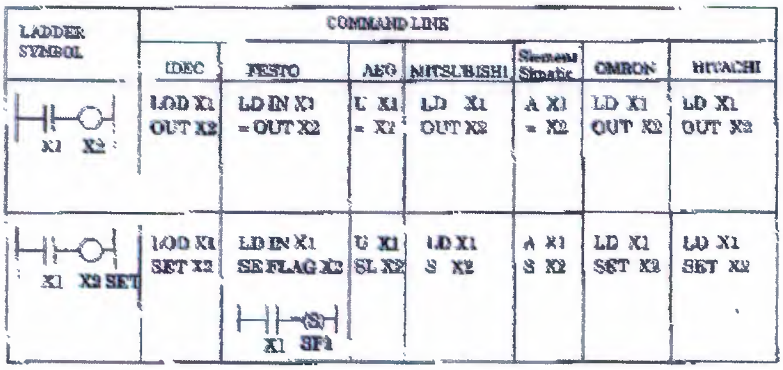
LADDER SYMBOL	COMMAND LINE						
	IDEC	FESTO	AEG	MITSUBISHI	Siemens SIMATIC	OMRON	HITACHI
X1 X2	LOD X1 AND X2	LD IN X1 AND IN X2	U X1 U X2	LD X1 AND X2	A X1 A X2	LD X1 AND X2	LD X1 AND X2
X1 X2	LOD X1 AND NOT X2	LD IN X1 AND NOT IN X2	U X1 UN X2	LD X1 ANI X2	A X1 AN X2	LD X1 AND NOT X2	LD X1 ANI X2
X1 X2	LOD X1 OR X2	LD IN X1 OR X2	U X1 O X2	LD X1 OR X2	A X1 O X2	LD X1 OR X2	LD X1 OR X2
X1 X2	LOD X1 OR NOT X2	LD IN X1 OR NOT X2	U X1 ON X2	LD X1 ORI X2	A X1 ON X2	LD X1 OR NOT X2	LD X1 ORI X2

Table 2.2.2: Symbol and command line AND and OR exercises.

c) Output Stored Exercises:

At a PLC system relay, it is used as output function, can be divided into two groups. First group output which charge can be linked to it according to program as (solenoid valves, neon lamb, conductor, led, etc.) are real output. Also second group outputs are internal and image relays. Physical connection cannot link to these relays but outputs of these sensors are transferred to real output and output can be taken.

If commands will be observed, there are similarities between PLC devices that output program commands are different. At both output and input functions, X1, X2, are used as addresses.

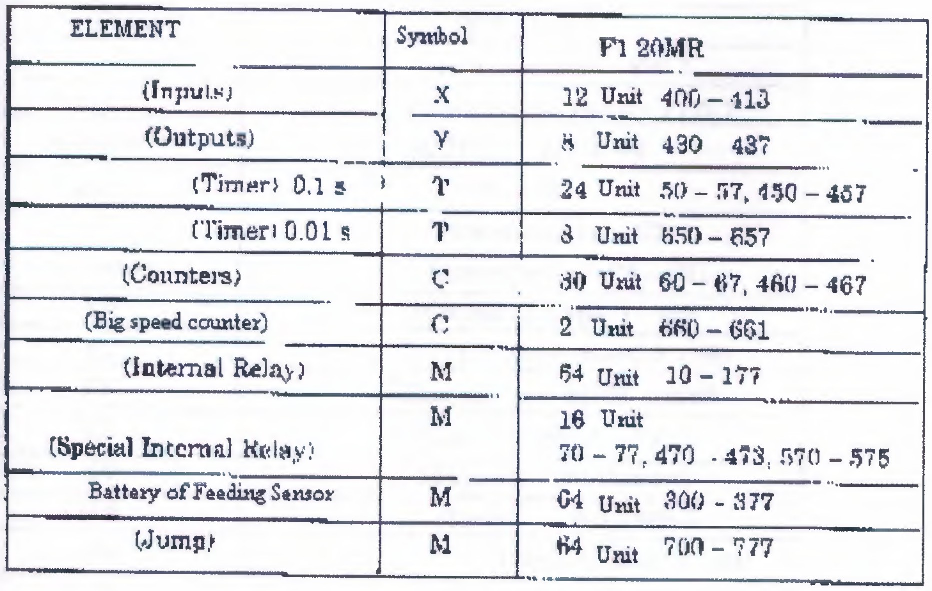


LADDER SYMBOL	COMMAND LINE						
	IDEC	FESTO	AEG	MITSUBISHI	Siemens Simatic	OMRON	HITACHI
X1 X2	LOD X1 OUT X2	LD IN X1 =OUT X2	U X1 = X2	LD X1 OUT X2	A X1 = X2	LD X1 OUT X2	LD X1 OUT X2
X1 X2 SET	LOD X1 SET X2	LD IN X1 SE FLAG X2 X1 SF1	U X1 SL X2	LD X1 S X2	A X1 S X2	LD X1 SET X2	LD X1 SET X2

Figure.2.2.3.

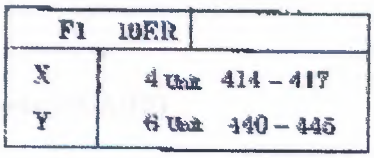
2.3. SPECIFICATION OF EXAMINED PLC

a) Mitsubishi F1 20 MR



ELEMENT	Symbol	F1 20MR
(Inputs)	X	12 Unit 400 – 413
(Outputs)	Y	8 Unit 430 437
(Timer) 0.1 s	T	24 Unit 50 – 57, 450 – 457
(Timer) 0.01 s	T	8 Unit 650 – 657
(Counters)	C	30 Unit 60 – 67, 460 – 467
(Big speed counter)	C	2 Unit 660 – 661
(Internal Relay)	M	64 Unit 10 – 177
(Special Internal Relay)	M	16 Unit 70 – 77, 470 - 473, 570 – 575
Battery of Feeding Sensor	M	64 Unit 300 - 377
(Jump)	M	64 Unit 700 – 777

Table 2.3.1: table of element and element numbers

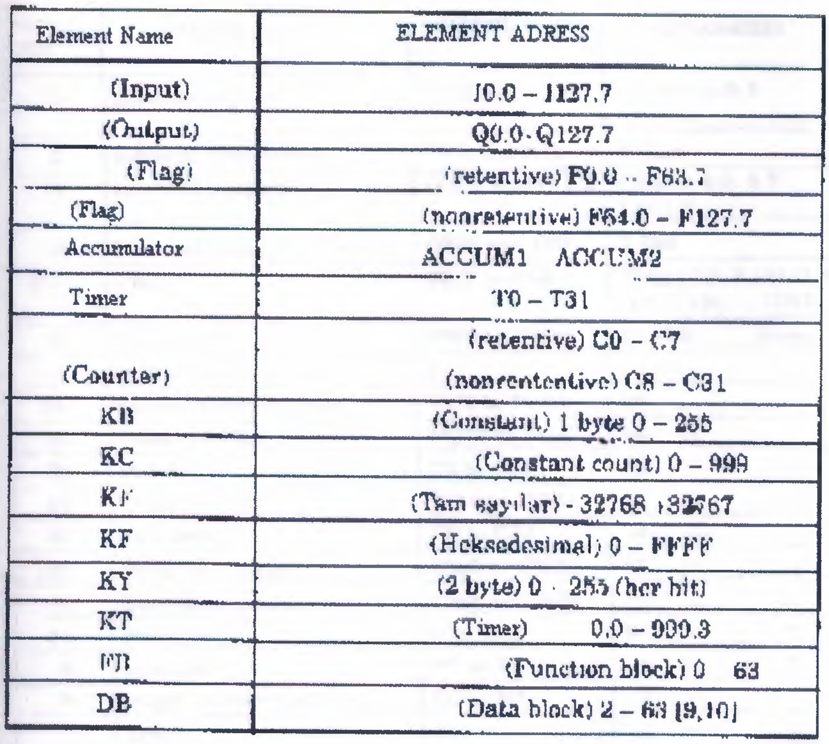


F1 10ER	
X	4 Unit 414 – 417
Y	6 Unit 440 – 445

Table 2.3.2. Increasing unit

F1 20 MR PLC as 12 inputs 8 outputs, which we use. If more input and output are necessary, input/output-increasing units are plugged to PLC. These units have various numbers output and input. At table 2.3.1, there are 4 inputs 6 outputs for F1 10 ER model.

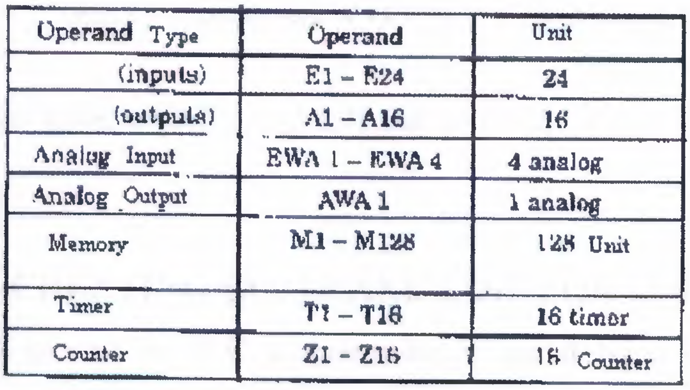
b) Siemens Simatic S5-90U



Element Name	ELEMENT ADRESS
(Input)	I0.0 – I127.7
(Output)	Q0.0-Q127.7
(Flag)	(retentive) F0.0 – F63.7
(Flag)	(nonretentive) F64.0 – F127.7
Accumulator	ACCUM1 ACCUM2
Timer	T0 – T31
(Counter)	(retentive) C0 – C7 (nonrententive) C8 – C31
KB	(Constant) 1 byte 0 – 255
KC	(Constant count) 0 – 999
KF	(Tam sayılar) - 32768 +32767
KF	(Heksedesimal) 0 – FFFF
KY	(2 byte) 0 - 255 (her bit)
KT	(Timer) 0.0 – 999.3
FB	(Function block) 0 63
DB	(Data block) 2 – 63 [9,10]

Table 2.3.3: Specifications of S5-90U model Siemens Simatic.

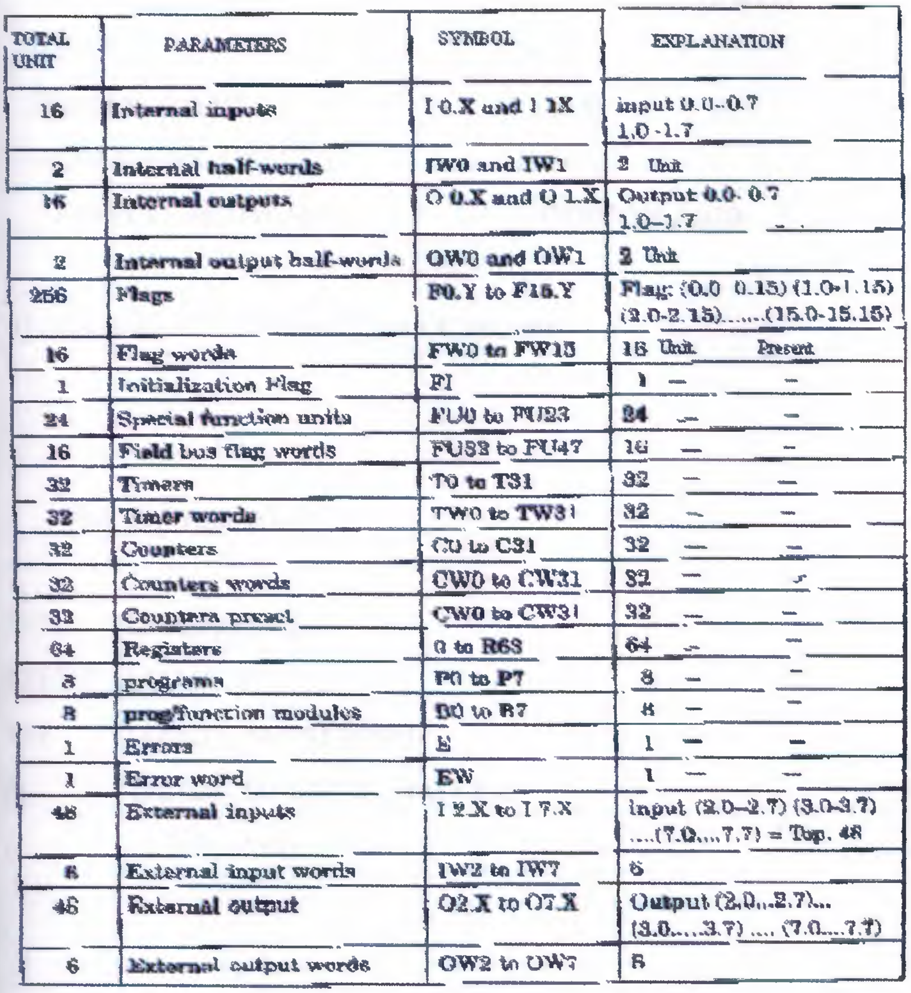
c) AEG Teachware modicon A020



Operand Type	Operand	Unit
(inputs)	E1 – E24	24
(outputs)	A1 – A16	16
Analog Input	EWA 1 – EWA 4	4 analog
Analog Output	AWA 1	1 analog
Memory	M1 – M128	128 Unit
Timer	T1 – T16	16 timer
Counter	Z1 - Z16	16 Counter

Table 2.3.4. Specifications of AEG Teachware A020

d) FESTO (FPC 202C)



TOTAL UNIT	PARAMETERS	SYMBOL	EXPLANATION
16	Internal inputs	I 0.X and I 1X	input 0.0–0.7 1.0 -1.7
2	Internal half-words	IW0 and IW1	2 Unit
16	Internal outputs	O 0.X and O 1.X	Output 0.0- 0.7 1.0–1.7
2	Internal output half-words	OW0 and OW1	2 Unit
256	Flags	F0.Y to F15.Y	Flag: (0.0 0.15) (1.0-1.15) (2.0-2.15)......(15.0-15.15)
16	Flag words	FW0 to FW15	16 Unit Present
1	Initialization Flag	FI	1 — —
24	Special function units	FU0 to FU23	24 — —
16	Field bus flag words	FU32 to FU47	16 — —
32	Timers	T0 to T31	32 — —
32	Timer words	TW0 to TW31	32 — —
32	Counters	C0 to C31	32 — —
32	Counters words	CW0 to CW31	32 — —
32	Counters preset	CW0 to CW31	32 — —
64	Registers	0 to R63	64 — —
8	programs	P0 to P7	8 — —
8	prog/function modules	B0 to B7	8 — —
1	Errors	E	1 — —
1	Error word	EW	1 — —
48	External inputs	I 2.X to I 7.X	input (2.0–2.7) (3.0-3.7) ....(7.0....7.7) = Top. 48
6	External input words	IW2 to IW7	6
48	External output	O2.X to O7.X	Output (2.0...2.7)... (3.0.....3.7) .... (7.0....7.7)
6	External output words	OW2 to OW7	6

Table 2.4.8 Specification of FESTO (FPC 202C) Module PLC

In this table, x=(0,1,2,3,4,5,6,7) and y=(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15) are.

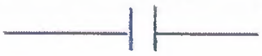
2.4. CREATING COMMAND LINE FOR LOGIC PROCESS

Each process in PLC programming is stated by a command and these commands provides connections of relay and contacts together, designations of outputs, counter, programming of timers and making of arithmetic comparison processes.

In our days, to experience PLC device of all firms are very hard. We will experience five brands. These brands are enough for us.

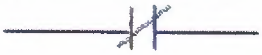
BRAND	**MODEL**
1) IDEC	FA1-JUNIOR (FA1J)
2)FESTO	202-C
3)MITSUBISHI	F120 R
4) SIEMENS-SIMATIC	S5-90U
5) AEG TEACHWARE	MODICON A020

a) Loading of Close and Open Contact:



Normally open contact

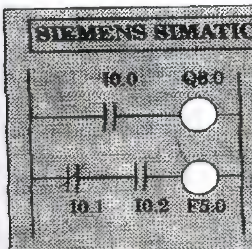
LOD	(LOAD)-IDEC
LD IN	(LOAD)-FESTO
LD	(LOAD)- MITSUBISHI
A	(AND)- SIEMENS-SIMATIC
U	(UND)-AEG



Normally close contact

LOD NOT	(LOAD NOT)-IDEC
LD NOT IN	(LOAD NOT)-FESTO
LDI	(LOAD INVERSE)- MITSUBISHI
AN	(AND NOT)- SIEMENS-SIMATIC
UN	(UND NICHT)-AEG

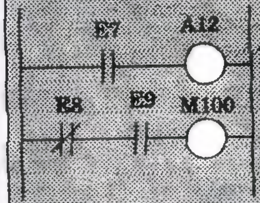
SIEMENS SIMATIC



```
: A   I 0.0
: =   Q 6.0
: AN  I 0.1
: A   I 0.2
: =   F 5.0
: BE
```

Input: I0.0–I127.7
Çıkış Q0.0–Q127.7
Flag F0.0–F63.7
BE, PE, END = PROGRAM SONU

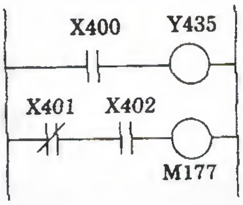
AEG



```
1 U    E7
2 =    A12
3 UN   E8
4 U    E9
5 =    M100
6 PE
```

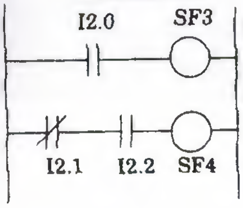
Girişler: E1–E24
Çıkışlar: A1–A16
Saklayıcılar: M1–M128

MITSUBISHI



```
0 LD   X400
1 OUT  Y435
2 LDI  X401
3 AND  X402
4 OUT  M177
```

FESTO



```
0000  LD   IN  2.0
0001   =  FLAG 3
0002  LD   IN  2.1
0003  AND IN  2.2
0004   =  FLAG  4
```

In here, commands for giving different brand and module normally. Explain to designation of contact and contact numbers are written after command.

In AEG and Siemens PLC, a load command is not used in Siemens Module, open contact command normally is load written A (AND), load process is relazing with AN (AND NOT) command.

In AEG module U (UND) and UN (UND NOT) commands are used for load process. As we know that these commands are used to serial AND and AND NOT exercises.

b) AND exercise:

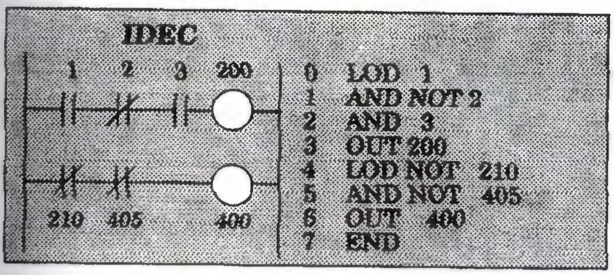
Serial contact linking commands

AND	-(IDEC)
AND IN	-(FESTO)
AND	-(MITSUBISHI)
A(AND)	-(SIEMENS-SIMATIC)
U (UND)	-(AEG)

c) AND NOT exercise:

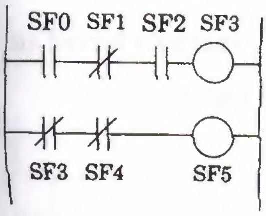
Serial contact linking commands

AND NOT	-(IDEC)
AND NOT IN	-(FESTO)
AND	-(MITSUBISHI)
A(AND)	-(SIEMENS-SIMATIC)
U (UND)	-(AEG)



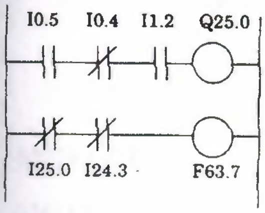
```
0  LOD  1
1  AND NOT 2
2  AND  3
3  OUT 200
4  LOD NOT  210
5  AND NOT  405
6  OUT  400
7  END
```

FESTO



```
0001  LD FLAG  0
0002  AND NOT FLAG  1
0003  AND FLAG  2
0004  = FLAG  3
0005  LD NOT FLAG  3
0006  AND NOT FLAG  4
0007  = FLAG  5
0008  LD PROG  0
```

SIEMENS SIMATIC



```
:  A   I  0.5
:  AN  I  0.4
:  A   I  1.2
:  =   Q  25.0
:  AN  I  25.0
:  AN  I  24.3
:  =   F  63.7
:  BE
```

d) OR exercise:

Parallel contact linking commands

OR	-(IDEC)
OR	-(FESTO)
OR	-(MITSUBISHI)
O(OR)	-(SIEMENS-SIMATIC)
O(ODER)	-(AEG)

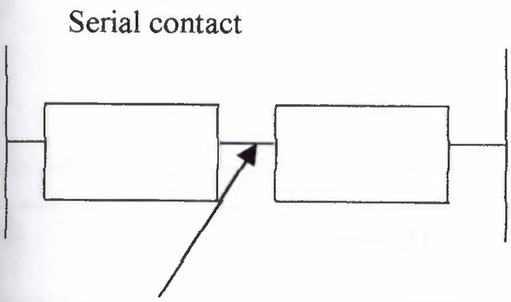
e) OR NOT exercise:

Parallel contact linking commands

OR NOT	-(IDEC)
OR NOT	-(FESTO)
ORI(OR INVERSE)	-(MITSUBISHI)
ON(OR NOT)	-(SIEMENS-SIMATIC)
ON(ODER NICHT)	-(AEG)

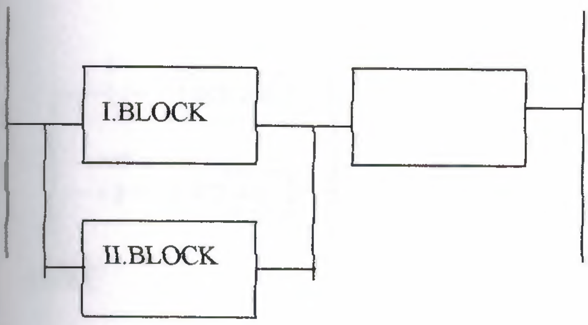
2.5. GET COMMUNICATE OF COMMAND BLOCK TOGETHER

a) Serial Contact:



AND LOD	-(IDEC)
AND LD	-(FESTO)
ANB (AND BLOCK)	-(MITSUBISHI)
A(.................	-(SIEMENS)
)	
U(.................	-(AEG)
)	

b) Parallel Contact:



OR LOD -(IDEC)

OR LD -(FESTO)

ORB (OR BLOCK) -(MITSUBISHI)

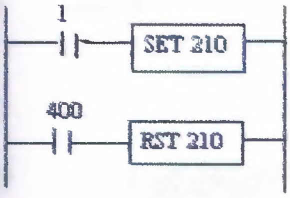
```
A(................        -(SIEMENS)
O
..................
)
```

```
O(                        -(AEG)
)
```

2.6 SET AND RESET INSTRUCTION

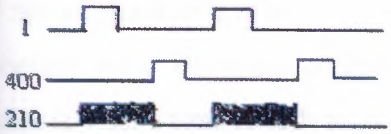
If any of the OFF position relay (eg. Input, output register and internal relay) pass the ON position that is from logic 0 to logic 1. Pass instruction called SET command. RESET command is opposite of SET command that is ON position to OFF position, from logic 1 to logic 0.

Another peculiarity of SET and RESET instructions for working instructions input must be control with relay. It does not require any continuos signal or stroke. That means SET relay always logic 1 position with input relay. If input relay done OFF position does not effect setted relay while that RESET command come.

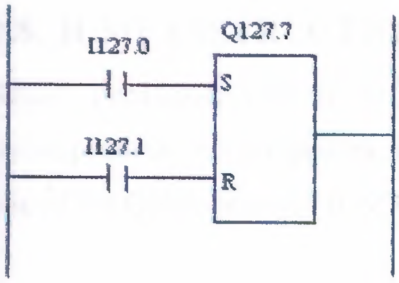


IDEC

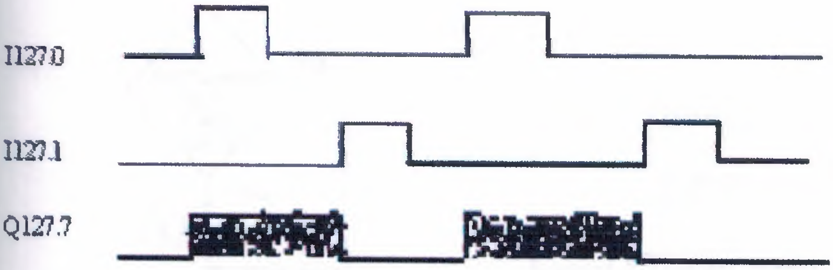
```
0        LOD    1
1        SET    210
2        LOD    400
3        RST    210
4        END
```



SIEMENS



```
A     I   127.0
S     Q   127.7
A     I   127.1
R     Q   127.7
BE
```



2.7.SINGLE OUTPUT INSTRUCTIONS

Our aim is make ON position, on scan time length. With these aim we use two different relays. First one is which makes control, other one is where we take output. The important point is; while controlling relay passing OFF position to ON, where output relay is 1 scan time length mould pass ON position to OFF. It is unimportant that controlling relay is protecting ON position. When the OFF position relay pass to ON position, we take 1 scan time length from output relay.

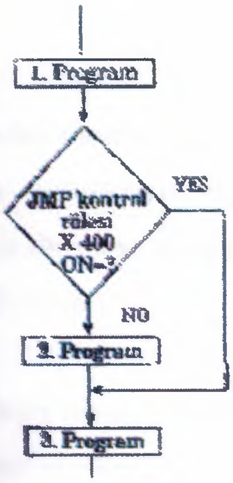
2.8. JUMP INSTRUCTION

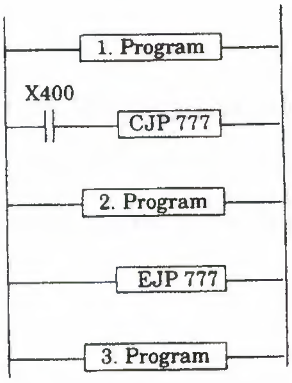
Source peculiarity with JUMP instruction; determined program line or lines makes possive position that jumped by some condition, or conditions. Provided jumped relay is time of the ON position of JUMP command.

MITSUBISHI

CJP (Conditional Jump)

EJP (End of Jump)





Note: JUMP instructions are between 700 – 777

Above program is between the 1. and 2. Programs because of using JUMP instruction, 400-numbered input relay when passed logic 1 position, JUMP instruction come to active condition and 2. program jumped 3. program, and 3. program started to work. Because after the EJP, JUMP ending operation instruction.

With 401 numbered input came logic 1 (ON) jumping operation starts and from CJP 700 until EJP 700 line program line jumps.

Jumping operation goes when X401 OFF. When X401 OFF done program return to work normally and scan operation works line by line.

While X401 OFF position, JMP function does not work. The important point is; before CJP instruction, EJP used must go to last EJP operation. Others will be invalid.

2.9 TIMERS

Let's now see how a timer works. What is a timer? Its exactly what the word says... it is an instruction that waits a set amount of time before doing something. Sounds simple doesn't it.

When we look at the different kinds of timers available the fun begins. As always, different types of timers are available with different manufacturers. Here are most of them:

On-Delay timer-This type of timer simply "delays turning on". In other words, after our sensor (input) turns on we wait x-seconds before activating a solenoid valve (output). This is the most common timer. It is often called TON (timer on-delay), TIM (timer) or TMR (timer).

Off-Delay timer- This type of timer is the opposite of the on-delay timer listed above. This timer simply "delays turning off". After our sensor (input) sees a target we turn on a solenoid (output). When the sensor no longer sees the target we hold the solenoid on for x-seconds before turning it off. It is called a TOF (timer off-delay) and is less common than the on-delay type listed above. (i.e. few manufacturers include this type of timer)

Retentive or Accumulating timer- This type of timer needs 2 inputs. One input starts the timing event (i.e. the clock starts ticking) and the other resets it. The on/off delay timers above would be reset if the input sensor wasn't on/off for the complete timer duration. This timer however holds or retains the current elapsed time when the sensor turns off in mid-stream. For example, we want to know how long a sensor is on for during a 1 hour period. If we use one of the above timers they will keep resetting when the sensor turns off/on. This timer however, will give us a total or accumulated time. It is often called an RTO (retentive timer) or TMRA (accumulating timer).

Let's now see how to use them. We typically need to know 2 things:

What will enable the timer. Typically this is one of the inputs.(a sensor connected to input 0000 for example)

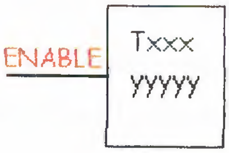
How long we want to delay before we react. Let's wait 5 seconds before we turn on a solenoid, for example.

When the instructions before the timer symbol are true the timer starts "ticking". When the time elapses the timer will automatically close its contacts. When the program is running on the plc the program typically displays the elapsed or "*accumulated*" time for us so we can see the current value. Typically timers can tick from 0 to 9999 or 0 to 65535 times.

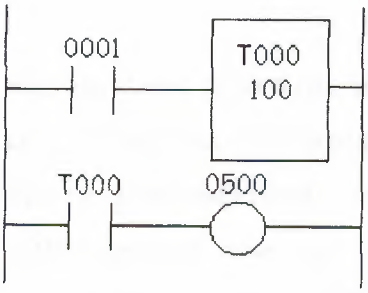
Why the weird numbers? Again its because most PLCs have 16-bit timers. We'll get into what this means in a later chapter but for now suffice it to say that 0-9999 is 16-bit BCD (binary coded decimal) and that 0 to 65535 is 16-bit binary. Each tick of the clock is equal to x-seconds.

Typically each manufacturer offers several different ticks. Most manufacturers offer 10 and 100 ms increments (ticks of the clock). An "ms" is a mili-second or *1/1000th* of a second. Several manufacturers also offer 1ms as well as 1 second increments. These different increment timers work the same as above but sometimes they have different names to show their time-base. Some are TMH (high speed timer), TMS (super high speed timer) or TMRAF (accumulating fast timer).

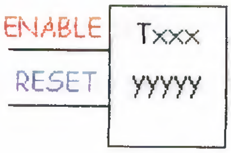
Shown below is a typical timer instruction symbol we will encounter (depending on which manufacturer we choose) and how to use it. Remember that while they may look different they are all used basically the same way. If we can setup one we can setup any of them.



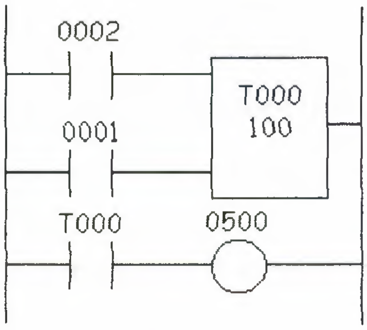
This timer is the on-delay type and is named Txxx. When the enable input is on the timer starts to tick. When it ticks yyyyy (the preset value) times, it will turn on its contacts that we will use later in the program. Remember that the duration of a tick (increment) varies with the vendor and the time-base used. (i.e. a tick might be 1ms or 1 second or...)



In this diagram we wait for input 0001 to turn on. When it does, timer T000 (a 100ms increment timer) starts ticking. It will tick 100 times. Each tick (increment) is 100ms so the timer will be a 10000ms (i.e. 10 second) timer. 100ticks X 100ms = 10,000ms. When 10 seconds have elapsed, the T000 contacts close and 500 turns on. When input 0001 turns off(false) the timer T000 will reset back to 0 causing its contacts to turn off(become false) thereby making output 500 turn back off.



This timer is named Txxx. When the enable input is on the timer starts to tick. When it ticks yyyyy (the preset value) times, it will turn on its contacts that we will use later in the program. Remember that the duration of a tick (increment) varies with the vendor and the time-base used. (i.e. a tick might be 1ms or 1 second or...) If however, the enable input turns off before the timer has completed, the current value will be retained. When the input turns back on, the timer will continue from where it left off. The only way to force the timer back to its preset value to start again is to turn on the reset input.



In this diagram we wait for input 0002 to turn on. When it does timer T000 (a 10ms increment timer) starts ticking. It will tick 100 times. Each tick (increment) is 10ms so the timer will be a 1000ms (i.e. 1 second) timer. 100ticks X 10ms = 1,000ms. When 1 second has elapsed, the T000 contacts close and 500 turns on. If input 0002 turns back off the current elapsed time will be retained. When 0002 turns back on the timer will continue where it left off. When input 0001 turns on (true) the timer T000 will reset back to 0 causing its contacts to turn off (become false) thereby making output 500 turn back off.

2.10. COUNTERS

A counter is a simple device intended to do one simple thing - count. Using them, however, can sometimes be a challenge because every manufacturer (for whatever reason) seems to use them a different way. Rest assured that the following information will let you simply and easily program anybody's counters.

What kinds of counters are there? Well, there are up-counters (they only count up 1,2,3...). These are called CTU,(count up) CNT,C, or CTR. There are down counters (they only count down 9,8,7,...). These are typically called CTD (count down) when they are a separate instruction. There are also up-down counters (they count up and/or down 1,2,3,4,3,2,3,4,5,...) These are typically called UDC(up-down counter) when they are separate instructions.

Many manufacturers have only one or two types of counters but they can be used to count up, down or both. *Confused yet?* Can you say "no standardisation"? Don't worry, the theory is all the same regardless of what the manufacturers call them. A counter is a counter is a counter...

To further confuse the issue, most manufacturers also include a limited number of high-speed counters. These are commonly called HSC (high-speed counter), CTH (Counter
High-speed?)

Typically a high-speed counter is a "*hardware*" device. The normal counters listed above are typically "*software*" counters. In other words they don't physically exist in the plc but rather they are simulated in software. Hardware counters do exist in the plc and they are not dependent on scan time.
A good rule of thumb is simply to always use the normal (software) counters unless the pulses you are counting will arrive faster than 2X the scan time. (i.e. if the scan time is 2ms and pulses will be arriving for counting every 4ms or longer then use a software counter. If they arrive faster than every 4ms (3ms for example) then use the hardware (high-speed) counters. (2xscan time = 2x2ms= 4ms)

To use them we must know 3 things:

Where the pulses that we want to count are coming from. Typically this is from one of the inputs.(a sensor connected to input 0000 for example)

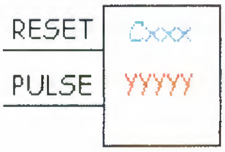
How many pulses we want to count before we react. Let's count 5 widgets before we box them, for example.

When/how we will reset the counter so it can count again. After we count 5 widgets lets reset the counter, for example.

When the program is running on the plc the program typically displays the current or "*accumulated*" value for us so we can see the current count value.

Typically counters can count from 0 to 9999, -32,768 to +32,767 or 0 to 65535. Why the weird numbers? Because most PLCs have 16-bit counters. We'll get into what this means in a later chapter but for now suffice it to say that 0-9999 is 16-bit BCD (binary coded decimal) and that -32,768 to 32767 and 0 to 65535 is 16-bit binary.

Here are some of the instruction symbols we will encounter (depending on which manufacturer we choose) and how to use them. Remember that while they may look different they are all used basically the same way. If we can setup one we can setup any of them.



In this counter we need 2 inputs.

One goes before the reset line. When this input turns on the current (accumulated) count value will return to zero.

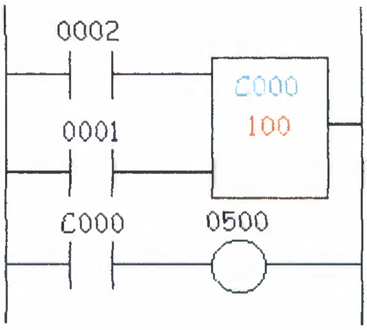
The second input is the address where the pulses we are counting are coming from.

For example, if we are counting how many widgets pass in front of the sensor that is physically connected to input 0001 then we would put normally open contacts with the address 0001 in front of the pulse line.

Cxxx is the name of the counter. If we want to call it counter 000 then we would put "C000" here.

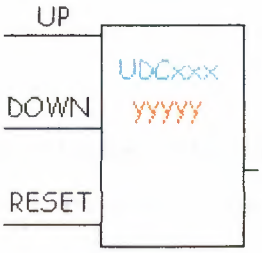
yyyyy is the number of pulses we want to count before doing something. If we want to count 5 widgets before turning on a physical output to box them we would put 5 here. If we wanted to count 100 widgets then we would put 100 here, etc. When the counter is finished (i.e we counted yyyyy widgets) it will turn on a separate set of contacts that we also label Cxxx.

Note that the counter accumulated value ONLY changes at the off to on transition of the pulse input.

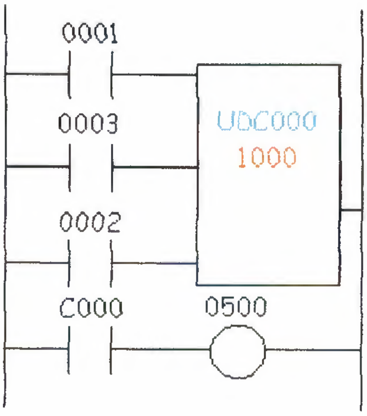


Here's the symbol on a ladder showing how we set up a counter (we'll name it counter 000) to count 100 widgets from input 0001 before turning on output 500. Sensor 0002 resets the counter.

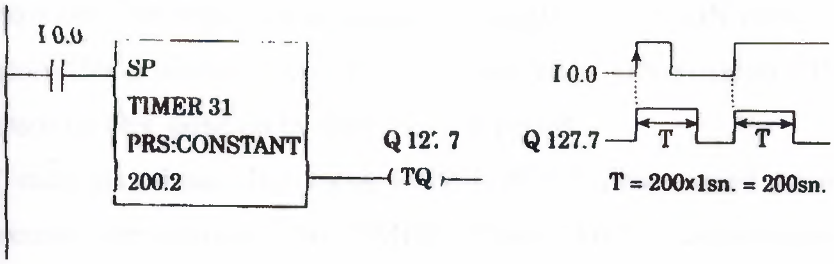
Below is one symbol we may encounter for an up-down counter. We'll use the same abbreviation as we did for the example above.(i.e. UDCxxx and yyyyy)



In this up-down counter we need to assign 3 inputs. The reset input has the same function as above. However, instead of having only one input for the pulse counting we now have 2. One is for counting up and the other is for counting down. In this example we will call the counter UDC000 and we will give it a preset value of 1000. (we'll count 1000 total pulses) For inputs we'll use a sensor which will turn on input 0001 when it sees a target and another sensor at input 0003 will also turn on when it sees a target. When input 0001 turns on we count up and when input 0003 turns on we count down. When we reach 1000 pulses we will turn on output 500. Again note that the counter accumulated value ONLY changes at the off to on transition of the pulse input. The ladder diagram is shown below.



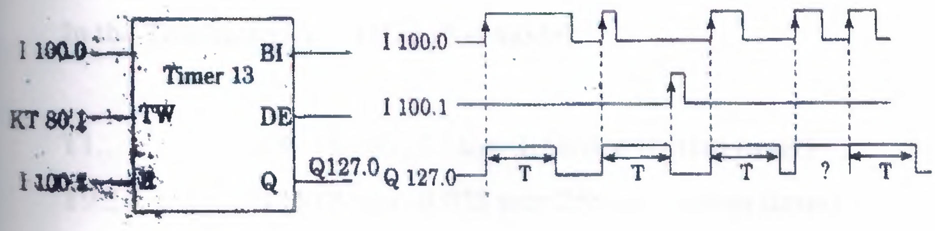
Siemens Simatic : Pulse Timer (SP)



10.0 input sensor works T31 timer. When this sensor takes ON position, settled till 200 sec, Q127.7 out put done 1. Even time over, if input signal I0.0 logic 1, output will reset.

```
:  A     I     0.0
:  L     KT    200.2
:  SP    T     31
:  =     Q     127.7
:  BE
```

Extended Pulse Timer



```
: A      I     0.0
: L      KT    100.2
: SD     T     12
: A      I     0.1
: R      T     12
: A      T     12
: =      Q     100.0
: BE
```

This kind of timer controls I100.0 input sensor 13 numbered TIMER. When I100.0 sensor was made 1, the sensor which was obliged Q127.0 numbered TIMER pass ON position. The important event is the pass of I100.0 to ON position not the time o this sensors ON position. Even I100.0 1msec stays ON position TIMER protects Q127.0 sensor on ON position by the time of T period.

T must stay 8 sec. But mean while I100.0 T time passed from logic 0 to 1 without second time charging. So TIMER output (Q127.0) protects its ON position again. But it returns beginning again to count from 0, of the T time.

```
:   A      I       100.0
:   L      KT      80.1
:   SE     T       13
:   A      I       100.1
:   R      T       13
:   A      T       13
:   =      Q       127.0
```

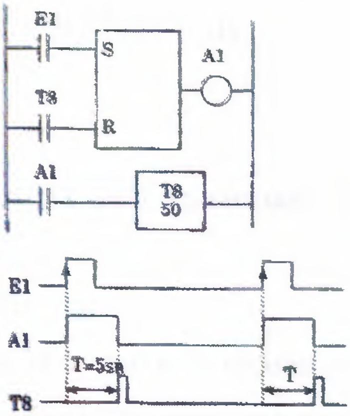
AEG

In the Teachware A020-020 Plus model;

T1....................T8 (8 unit, 0.1sec=100msec rhythm timer)

T9....................T16 (8 unit, 0.025 sec=25msec rhythm timer)

In order to 16 unit (T1.............T16) TIMER there are so programs be smallest and biggest time value is 25 msec which is 110 minutes.



```
1    U     E1
2    SL    A1
3    U     T8
4    RL    A1
5    U     A1
6    =     T8
7          50
8    PE
```

In this example A1 is stetted with E1 output. Reset position is the time of, when T8 pass ON position.

When E1 pass ON position A1 output makes set. By the setting of A1,T8 timer (present value 50x 0.1sec=5sec) count in its inside 5sec and at the end of this time logic done 1. As to program; when T8 is on,A1 output makes resent and T8 output goes OFF position because T8 output is armed reset sensor. The event to care on TIMER present value; chosen TIMER's rhythm times by its number, because of its changes, present value must count right.

The program on above; 413 numbered input sensor and M73 numbered private internal sensor are used to reset 467 numbered counter. Counting input is controlled by 412 numbered input sensor. Present value of counter is showed with K20-20. The input of counter pulse's every present pulse value is lowered 1 degree.

2.11. SHIFT REGISTER

IDEC

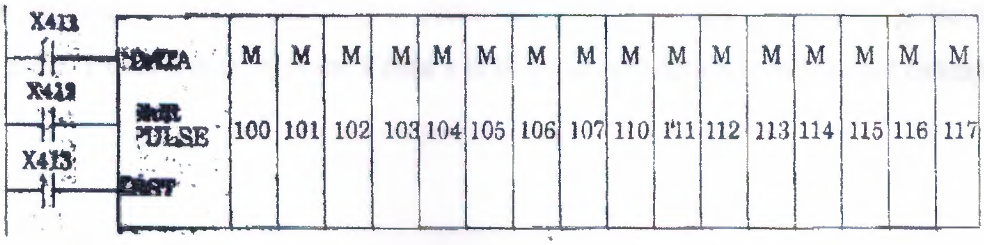
This model in PLC shift register unit has studied extensively.

MITSUBISHI

Internal relay M is used shift register at the some time. So 16 sensor must be 1 group at the same time First helping sensor number, shift register address and following 16 sensor can not use another arm.

Shift Register Addresses

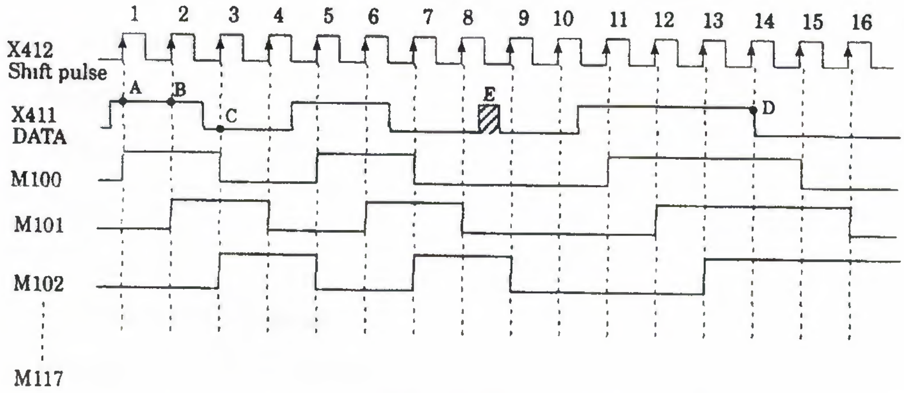
M100 – M117	=	M100..........M107,	M110..........M117 = 16 unit
M120 – M137	=	M120..........M127,	M130..........M137 = 16 unit
M140 – M157	=	M140..........M147,	M150..........M157 = 16 unit
M160 – M177	=	M160..........M167,	M170..........M177 = 16 unit
M200 – M217	=	M200..........M207,	M210..........M217 = 16 unit
M220 – M237	=	M220..........M227,	M230..........M237 = 16 unit
M240 – M257	=	M240..........M247,	M250..........M257 = 16 unit
M260 – M277	=	M260..........M267,	M270..........M277 = 16 unit
M300 – M317	=	M300..........M307,	M310..........M317 = 16 unit
M320 – M337	=	M320..........M327,	M330..........M337 = 16 unit
M340 – M357	=	M340..........M347,	M350..........M357 = 16 unit
M360 – M377	=	M360..........M367,	M370..........M377 = 16 unit

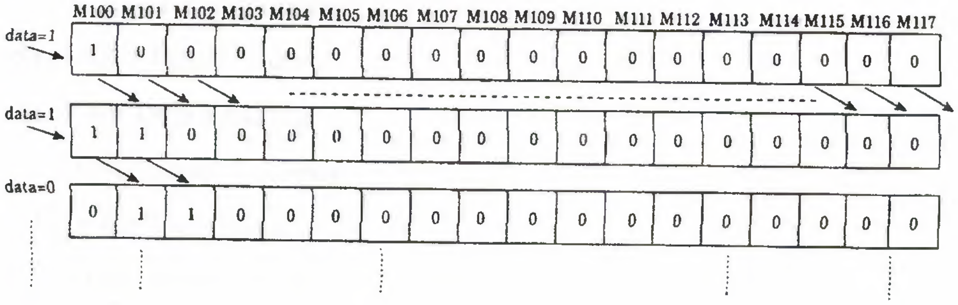


1-Data input: Data signal which must be given to Register, is designed ON-OFF position to X411 sensor. Data, entered to register, firstly apply to M100 register. But every shift operation can make by shift pulse.

2-Shift pulse: It is shift input which is transferred to M100 by X411 entered data but while X412 is passing from 0 to 1. It can be used 72 numbered which produces 100msec time pulse or 73 numbered which produces on msec time pulse generator instead of X412.

3-Reset input: X413 input sensor is used for reset of the above. So all the register sensor with X413's passing OFF position to ON position makes reset and pass of position (M100.................M117).





	M100	M101	M102	M103	M104	M105	M106	M107	M108	M109	M110	M111	M112	M113	M114	M115	M116	M117
data=1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
data=1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
data=0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(1100110000111100) data is applied with X411 data input on the above example. In here the important thing is decisive position of data m the shift pulse time. For example, 1 data's is in A point 1 data's is in B point 0 data's is in C point examples.

Decisive position in D point is 1, because while shift pulse going from 0 to 1; data value stayed decisively periods 1 pulse time in ON position, so D point of data's the time of going from 1 to 0, shift pulse which is still formed, can't catch and it can't be seen and examples the time of going from 1 to 0 of 14 pulse.

If you attend E area of data diagram; it can't be exampled by data which is between 8 and 9 pulse and it doesn't accept like this data. According to this, for to load of data's to registers is the time of passing the time of piece of referans shift pulse (OFF→ON)

2.12. COMPUTING FUNCTION

one of the most important peculiarity of PLC system is computing and data embroidery function. As a main structure, PLC has this peculiarity.

Some of these are:

1) Addition
2) Subtraction
3) Multiplication
4) Division
5) BCD Binary Converter
6) BINARY BCD Converter
7) 4 DIGIT Comparation
8) 16 Bit Data Loading
9) 8 Bit Data Loading
10) Data Saving-Decrease
11) 16 Bit Data Store
12) 8 Bit Data Store
13) Data Display
14) BCD Shifting Left
15) Data Shifting

SIEMENS (Simatic) Comparison Function

In comparison operations:

!=F (equal)

≠F (not equal)

>F (big)

>=F (big equal)

<F (small)

<=F (small equal)

Instructions are used for make desired comparison, and if YES decision is reached, Q output will give ON position >F control was done at the above. According this, IB0 value which is in ACCU2 will be compared with IB1 in ACCU1, if ACCU2>ACCU1, Q100 will remove ON position. If this condition is not provide, Q100 will stay OFF position.

Arithmetically +F instruction will provide addition of 2 complete number this instruction add ACCUM1 and ACCUM2, of for –F instruction distinct the 2 number.

From ACCU2's contents will distinct ACC1's contents.

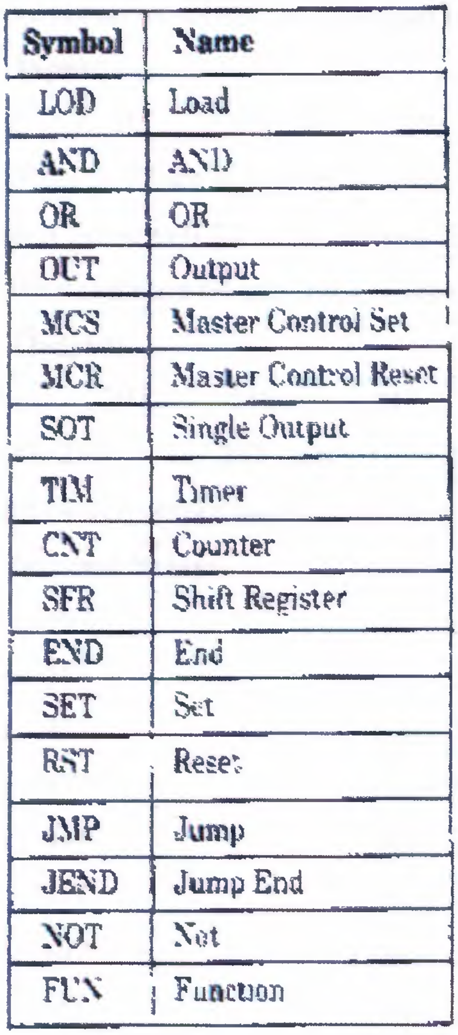
SECTION III

DETAIL ANALYSIS OF PROGRAMMING

3.1 BASIC INSTRUCTION WORD

Instruction word list

a) Basic Instructions:



Symbol	Name
LOD	Load
AND	AND
OR	OR
OUT	Output
MCS	Master Control Set
MCR	Master Control Reset
SOT	Single Output
TIM	Timer
CNT	Counter
SFR	Shift Register
END	End
SET	Set
RST	Reset
JMP	Jump
JEND	Jump End
NOT	Not
FUN	Function

b) FUN (Function) Instructions:

We can divide the instructions into 2 parts. These are ;

One – address instruction

Two – address instruction

There are 2 kinds of address instruction. Generally first address is the instruction word. In LOD, AND, OR, OUT, SET, RST, SOT instructions; there is a instruction word and number and addressing is obstructed with this that single addressed instruction.

Two addressed instructions; SFR, SFR NOT, TIM, CNT, FUN 100-146, FUN 200-246, TIM FUN, CNT FUN, FUN 147 and FUN 300. In this instructions first addresses are give instruction word and instruction numbers (Except FUN 147, FUN 300). As for second addresses are present peculiarity according to instruction.

There are some deliver numbers that referenced by FA1J at the below.

c) Input:

0...........7, 10...........17, 20..........27, 30...........37, 40...........47, 50...........57, 60..........67, 70..........77 are numbered like this. In here inputs are considered to OCTAL system which is between 0-77. If you attend 8,9,18,19,28,,29,...........78,79, numbers are not used. In octal there are 64 unit input number between 0-77 (except 8 and 9).

d) Output:

200...........207, 210............217, 220.............227, 230.............237, 240............247, 250.............257, 260.............267 and 270..............277 numbered. Like input there are 64 unit output numbers between 200-277 (except 8 and 9).

e) Internal Relay:

400 – 407
410 – 417
420 – 427
430 – 437
440 – 447
450 – 457
460 – 467
470 – 477
480 – 487
490 – 497

500 – 507
510 – 517
520 – 527
530 – 537
540 – 547
550 – 557
560 – 567
570 – 577
580 – 587
590 – 597

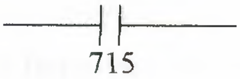
600 – 607
610 – 617
620 – 627
630 – 637
640 – 647
650 – 657
660 – 667
670 – 677
680 – 687
690 – 697

There are 240 units (30x8=240) internal relays between 400 and 697, we can appoint the TIMER, COUNTER or FUN outputs to the any of 240 sensor and then can use of this sensor for take new data or count value.

f) Special Internal Relay:

There are 16 units become 700-707 and 710-717. As an example of these, we can use the signal generator which produces 1 sec clock sign, that means we can use 1 Hz clock pulse sing ready.

We can use the signal generator which produces 0.1 sec clock sign that means 10 Hz clock pulse sign ready.



g) Timer:

There are totally 80 unit timers between 0 and 79. If you attent you can use 8 and 9. You can use any of TIMER that include 0 and 79. In there its enough to know for you that totally there are 80 unit TIMER that include 0-79.

h) Counter:

Totally there are 45 unit counter between 0 and 44. If you attent you can use 8 and 9.

i)Reversible Counter:

It is counter which can be counted forward or review. While other counters can only count forward counters number 45-46 can count forward or review. Counter 45 has up and down pulse input edge yet counter 46 is connected to only one input of up/down situation and when this edge is 1 up and when it be comes 0 it counts down.

j) Shift Register:

There are 128 shift register between 0 and 27 including 8-9.

k) Single Output:

We can use 96 SOT functions between 0 and 95 including 8-9.

l) Data Register:

Between DRO and DR99 and between 800 and 899, we have 100 data register.

3.2 FA1J SERIES ALLOCATION NUMBERS OF SPECIAL RELAYS

As known special relays are 700 and 717 relays except 708 and 704 from these numbers 700 and 705 are unused.

701 and 702 Start Control: When input number 0, which used to start the program is on or if number 500 has been appointed to automatic start process. It starts to turn the program on. Special relays 701 and 702 are off the process of the program is stopped.

703 All Output OFF: All outputs between 200 and 277 are off when special relay 703 turns into ON.

704 Initialize Pulse: Special flag (1 scan time) 704 becomes on as much as the time equalling 1 scan time. When program FA1J started being processed.

704 Numerical Value Error: Is there an error in computing instructions results. 706 becomes on for example; if the result of a subtraction process is lower than –10.000, special relay 706 becomes on. They make sure that the program is correct from the point of view numerical process while they register the programs.

707 Curry and Borrow: It there is carry or borrow in the results at computing instructions. 707 is set for example; in a addition process the total of 2 numbers are higher than 9999,707 is on.

713 1 sec. Timer Reset: When 713 is on special relay 714 is always reset mode.

714 1 sec. Clock: It is possible to take signal generator producing clock sign for one second or clock pulse sign for 1 Hz from special relay 714.

715 100-msec. Clock: We can remove our clock pulse that is for 10 speed by using special relay output of 715 with this sign.

716 Timer/Counter Preset Value Changed: Special relay 716 becomes on when timer counter preset value has been changed into unit of FA1J CPU. It is possible to delete 716 when pressed key of TR S, ENTR and ENTR. If a program is registered in memory.

717 In-operation Output: Relay 717 is always on while FA1J is operating of the program has ended this relay becomes off.

3.3. BASIC INSTRUCTION

Each program written in PLC are started in 2 ways. One at these that we can draw the program with its symbols in the location called Ladder Diagram and load it to the computer as this. The second one is that we can make direct attribution using the key team of PLC. Because of this it will be told example symbol and attribution us. Instructions later whole LOD instruction and the other instructions are being stated.

a) LOD Instructions:

This instructions is used at the beginning of logic diagram lines. It can be used once back by back or more than once to determine the situation at the beginning of the instructions such as AND LOD, OR LOD, SFR, CNT, TIM. As you see below an input relay is wanted to be loaded as a program. Symbol of it is declared as a show in ladder diagram. Program list from the statement.

This program is loaded as 0 LOD 1 and 0 which is seen an address must be given in each line of the end one by one starting from each line of the program. Value is appointed to each line orderly. We have mentioned before which numbers are separated for shift register, output, input, special relay, timer counter. Imaginary internal relay at the machine PLC.

We can divide our load process into 4 groups according to our functions.

b) Input, Output, Internal and Special Relays:

In the examples above example relay circuit of relay in ladder diagram and how the process of key and as a result of this the format seen in deplay was given.

- We can choose a value between 0 and 77 except 8 and 9 in the example of input.
- We can choose a value between 200 and 277 except 8 and 9 in the example of output.
- We can choose a value between 400 and 697 except 8 and 9 in the example of internal relay.

You can use special relay which you need are between 700-717 in the example of special relay for example I use pulse generator of clock for one speed with special relay 714.

c) Timer:

I wanted to use T8 timer from the 80 timers between 0-79 including 8 and 9 here and you see how the load process had been done.

d) Counter:

You can use any counter between 0 and 46 including 8 and 9. Load process is the same as aside.

e) Shift Register:

You can use any register from 128 of them between 0 and 127 including 8 and 9. Shift register numbered 1 was loaded in the next side.

f) AND Instruction:

It is same as AND logic we studied in Logic lessons. Both keys that are connected each other rapidly are on, output is on and is the other situations it becomes OFF in logic. In a multiplying processes both inputs are 1 than output is 1. And had it ended with 2 limit switches and 1 solenoid valve in order to understand the logic better. In diagrams, it is stated as relay ladder diagram and logic diagram. So we can tell that LS1 relay A and LS2 relay is B input and output is Y. In such equality it is that Y=A.B according to the compulsion of Boolean. If both inputs are 1 (ON) Y output will be ON. In other 3 probabilities, output Y will be 0 (OFF). You can see this in the table of truth.

As known, the series of TTL is Logic entegrate containing 4 and gate with 2 inputs in 7408. As in the circuit ¼ has been made equal to ladder diagram by using 7408. In both of them the function of output and working are same.

g) OR Instruction:

Or instruction has the same functions as or gate logic we studied in logic lessons. In here, just only one of the keys are OFF or 1 is enough for output to be 1 as 2 keys are connected in the parallel way. As a result there is addition process and in this process one of the 2 parallel inputs is enough to be one. I gave 2 important information's with or instruction. One as them is out function that is symbolised with 200 in the circle. I will speak about out function 2 or 3 classes later. But now, I gave output of parallel circuit, output 200 for the first time it means that: I mentioned that special relay 704 is a clock pulse generator that has f=1 Hz. You see signal of clock pulse in the diagram. We determined time of 1 and 0 in input relay of 36 by chance now so that nothing will be by chance in the following lessons. Let's accept that there is a time diagram for to learn or let's assume that input 36 is gained by making ON/OFF in the form. If we think that output 200 is connected to a lamb, the situations that lamb will be on are the times that output 200 is 1.

In this example, in order to understand or instructions better firstly, 2 limit switches were connected to each other rapidly and shown a ladder diagram and a solenoid valve control in output of it. And same circuit has been gained Logic equality by using only 1or gate of integrate of 7432. It is enough to make on only one of the inputs for the outputs to be ON in 3 equaliavence circuit to make output OFF. It is necessary to make both parallel inputs OFF. This position was shown in the truths table below.

h) NOT Instructions:

It has the same duty as NOT gate that you studied in the logic lessons. We take the opposite of the sign. If we have a look of the example above, they take the opposite of input relay 1 in PLC. If you carry out 1 logic level to input 1 from the outside, the sign is going to continue from B point as logic 0, because of the instruction of LOD NOT 1.

As it can be seen in time diagram, LOD NOT 1 instructions got the opposite of input 1. It is symbolised

i) AND LOD Instruction:

It is one of the most important of the basic instructions. You have to understand it very well AND LOD instruction, as shown a side is used to connect the instructions to each other in two different blocks. There can be some instructions lines that were mode with or, and, not, input shown before you in the blocks staled before. Watch out that in the programming. Thinks as if you were opening a ladder wall or parentheses. When you are beginning a new block. In each block open the relays and each new block with LOD. If you have write the program and at lost connect the both blocks with AND LOD.

j) OR LOD Instruction:

If you got AND NOT instruction well. It is easy to comprehend OR LOD. As sign in the ladder diagram above, 2 different blocks were connected rapidly this time. PLC this operator in its own memory, it makes it like this in operation register and stock register. There are 1 operation register and maximum 8 stock register that we used to make temporary loading in PLC. Operation register is the register that procedure is mode stock register is assistant register. We have maximum 8 stock register. If we load the I. Block, these instructions are loaded to the register. While II. Block is being loaded. I. Block in operation register is slide to assistant register. Now II. Block was set to operation register and I. Block to stack register. II. Block are connected to each other in a parallel way with OR LOD instruction given later.

SECTION IV

SIEMENS SIMATIC S5 95U

General Information

The Weighing and Force Measuring System SIWAREX P (type 7MH4205 - 1AC01) can be integrated into decentrally structured automation concepts via the SIMATIC S5-100U bus.

With automation device type S5-95U or S5-100U up to four and with type S5-90U up to three SIWAREX P can be addressed (slot 0,2,4,6).

SIWAREX P can for example be linked with SIMATIC S5-115U and up to S5-155U via the decentralised periphery system ET200 (connection IM318-B and IM 308-B).

In this way up to eight SIWAREX P can be addressed with one connection (IM318-B).

The enclosed handling module (order No. 7MH4811-5AP41) carries out the cyclic reading of all values of the SIWAREX P weighing system into the automation device.

4.1 TELEGRAM STRUCTURE:

Amount of data	Telegram identification: 5	Format: KF
Error status	Telegram identification: 7	Format: KM
Parameterization status	Telegram identification: 9	Format: KM
No. Of pos. Behind decimal point	Telegram identification: 11	Format: KH
Measured value	Telegram identification: 13	Format: KF
Limit value 1	Telegram identification: 15	Format: KF
Limit value 2	Telegram identification: 17	Format: KF
Dead load	Telegram identification: 19	Format: KF
Analog end value	Telegram identification: 21	Format: KF
Nominal load	Telegram identification: 23	Format: KF
Characteristic value	Telegram identification: 25	Format: KF
Limit frequency	Telegram identification: 27	Format: KF

Scaling status Telegram identification: -- Format: KM

The measured value is read cyclically.

4.2 MODULE DESCRIPTION:

File "SIWAP1ST.S5D" contains the following modules:

- **FB63**	for S5-90U; S5-100U (CPU 100/102) uses the scrawl markers MB 120-MB123
- **FB226**	for S5-95U; S5-100U (CPU 103) uses the scrawl markers MB 250-MB253
- **OB1**	example for S5-90U, S5-95U, S5-100U (CPU 100-103)

Caution! Prior to transferring the modules into the automation device (see item 1.6).

4.3 STORAGE REQUIREMENTS:

A minimum storage capacity of approx. 0.260 kilo-words is required.

4.4 PARAMETERISATION:

For parameterisation, the hardware address has to be entered into the scaling DB in data word 0. Several examples of this are included in the scope of delivery

(DB63, DB62, DB199). With SIWAREX P, KF64 in DW 0 corresponds with slot 0.

Further possible slots respectively hardware addresses are 80, 96 and 112.

If more than one SIWAREX P is used in an automation device, then the scaling DB has to be entered for the additional scale. For each scale, FB226 respectively FB63 has to be cyclically called with the corresponding scale DB.

4.5 DATA BLOCK:

DW 0 is pre-allocated (e.g. KF 64 = slot 0). In DW 2- 14 the SIWAREX P data is stored cyclically.

DW 0 KF = +00064

hardware address

DW 1 KY = 000,000	telegram identification/step counter
DW 2 KF = +00000	amount of data
DW 3 KM = 00000000 00000000	error status
DW 4 KM = 00000000 00000000	parameterization status
DW 5 KH = +00000	no. of pos. beh. decimal point
DW 6 KF = +00000	measured value
DW 7 KF = +00000	limit value 1
DW 8 KF = +00000	limit value 2
DW 9 KF = +00000	dead load
DW 10 KF = +00000	analog end value
DW 11 KF = +00000	nominal load
DW 12 KF = +00000	characteristic value
DW 13 KM = 00000000 00000000	status byte
DW 14 KF = +00000	limit frequency filter

Simatic S5 PLC Range

Siemens PLCs are widely used around the world and are available in many configurations and sizes. Bytronic have selected the S5 range of PLCs and applied them to many of the products available in this catalogue. The S5-90U is the standard 16 I/O unit for use with units requiring digital signals only. The S5-95U and 100U both have additional analogue capabilities for the more sophisticated products such as the Enhanced PLC Trainer (ICT3GH) and Process Control Unit (PCU3). As always, all PLCs are available in a pre-wired PLC Mounting Frame.

O RDER CODE	B ASE UNIT	NPUTS	OU TPUTS	A NALOGU E	R ANGE	M EMORY
P LCS1 PLCS2 PLCS3	S 5-90U S5-95U S5-100U	 0 16 16	6 Relay 16 Transistor 8 Relay	- 8 Inputs 1 Output 4 Inputs	- 0-10V 0-10V or 0-20mA 10V	2 K EEPROM 2K EEPROM 4K EEPROM

If you wish to order a PLC on a Mounting Frame, please add "F" to the code

S5 Accessories:

The S5 series of PLCs has a number of programming options available. The PC programming software follows the Siemens STEP5 structured programming which divides the programming into manageable pieces where only appropriate program blocks are executed to increase scan times. The software is available in an unlicensed format (PLCSP6) for the S5-90U PLC, while the S5-95/100U PLCs require a donglised version (PLCSP7). Both packages include the PC interface cable. Alternatively , the Siemens POG605U Programming Panel (PLCSP1) is a small, on-line LCD programming panel suitable for entering basic programs into any of the S5 range of PLCs. Programs are entered using the Statement List (STL) method and programming is aided by the display of system prompts and error messages.

Drivers:

The following is a list of drivers in the libraries for Siemens S5 PLCs. These drivers were prepared with Siemens Simatic S5 Step5/ST version 7.02 programming software. The single channel input and single channel output drivers are available on the Driver Installation disks. All other drivers are available on our web site. They contain rung comments and address symbols that are viewable in the programming package and on printouts.

DRIVER	DESCRIPTION
Msp_I	1 Channel Input
Msp_I_M	2 Channel Input Multiplexed
Msp_I_H	1 Channel Input High-Speed Counter
Msp_O	1 Channel Output
Msp_O_M	2 Channel Output Multiplexed
Msp_IO	1 Channel Input, 1 Channel Output
Msp_IO_M	2 Channel Input Multiplexed,2Channel Output Multiplexed

1- **Memory Usage:** The attached table shows the drivers that are available for Siemens S5-95U and their memory usage.

2- **Other Models:** The drivers should work any of the S5 processors that support the following two features

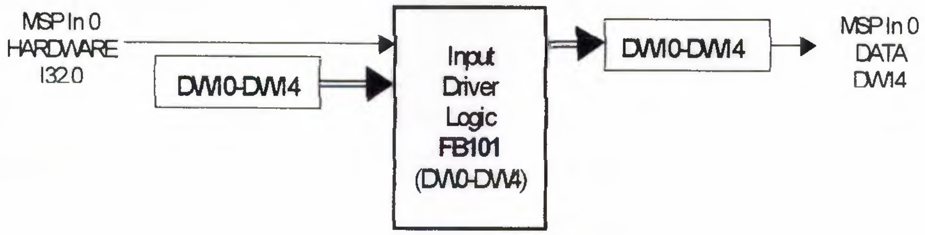
a) time-controlled program

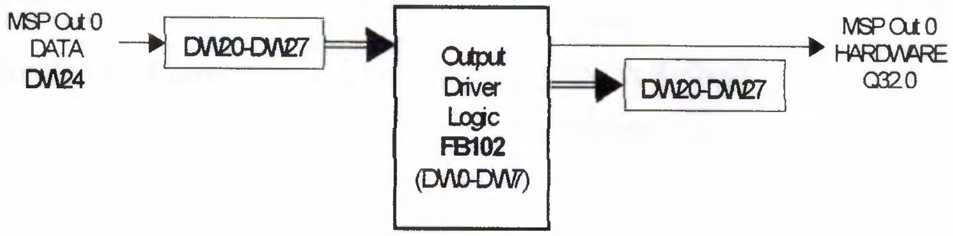
b) function blocks

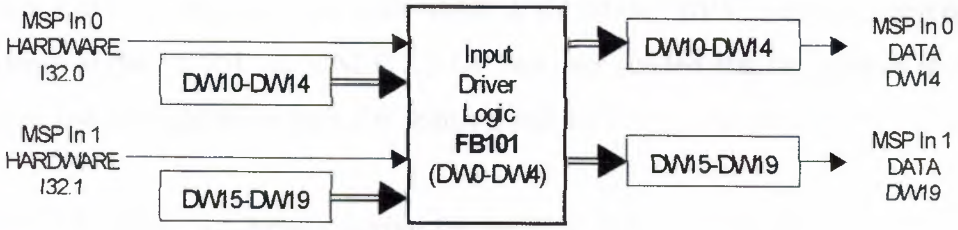
The S5-95U is an example of a model that does have these features. The S5-90U is an example of a model that does not have these features.

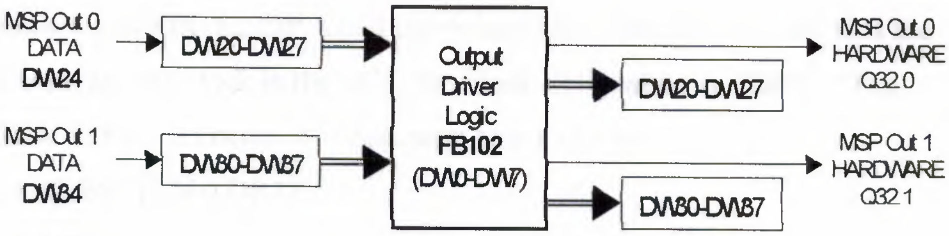
a- **Block Diagram.** The following block diagrams show the flow of execution and the data resources used by the different drivers

All DW memory locations are in DB2









b-Timing Parameters. These drivers are set up for the Delta protocol. Following are some of the key timing parameters:

Input

Protocol	Delta
Scan Time	20 msec
Full Word Bits	16 Bits
Delta Bits	4 Bits
Delta Refresh Count	16 Scan Refresh
ID Pulse Width	1.2 Scans

Data Pulse Width	3 Scans

Output

Protocol	Delta
Scan Time	20 msec
Full Word Bits	16 Bits
Delta Bits	4 Bits
Delta Refresh Count	16 Scan Refresh
ID Pulse Width	3 Scans
Data Pulse Width	3 Scans

In the DB1 parameter block with block designation "DPS" you parameterize the user address of the S5-95U as SINEC L2-DP bus user (called station number in the following) and the configuration data for sending and receiving via the SINEC L2-DP interface.

In the DB1 there is a default setting for the SINEC L2-DP interface.

A default DB1 is integrated in the operating system of the S5-95U programmable controllers; among other things it includes the preset parameters for data communications via SINEC L2-DP. Load the default DB1 into your PG and find the SINEC L2-DP parameter block in the DB1, the block designation is: "DPS:". The parameter block "DPS:" is enclosed in comment characters (#):

156: KC =' ; # #DPS: TLN 0 DPAE '

168: KC =' ; # END ';

The parameter block is not interpreted by the PLC in this form. Therefore you must overwrite with a blank the comment character directly in front of the block designation ("DPS:") and the comment character after the last SINEC L2-DP parameter:

156: KC =' ; # DPS: TLN 0 DPAE '

168: KC =' ; END ';

Transfer the changed DB1 to the S5-95U; by doing so you overwrite the default DB1.

After the STOP>RUN transition the S5-95U adopts the changed parameters.

As an example, the AG95U-DP is to be configured with station number "4" (L2 address) and a module with 16 words DI (0 to 31; consistency: "total length", identification byte: "223") and a module with 16 words DO (0 to 31; consistency: "total length", identification byte: "239"). The data from STEP7 for the DB1 is thus as follows

156: KC =' ; # DPS: TLN 4 DPAE ' ;

168: KC =' 223 239 ; END ';

An address ID must be entered behind "DPAE" for each module. Address IDs are as follows from STEP 7: for 16 words consistent DI data (223) and for 16 words consistent DO data (239). You must enter the address IDs in exactly this order in the DB1.

After address assigning in the S7 tool HWCONFIG you must enter the address ID received from the system into the DB1 of the S5-95U.

Below is a table with an example of codes of the send and receive data for S5-95U as DP slave.

Structure of the configuration byte:

Bit 7	**Bit 6**	**Bit 5 and 4**	**Bit 3, 2, 1 and 0**
128	64	32 16	8 4 2 1
Consistency	Length format	Send or receive data	Length of data
0: byte or word 1: total length	0: byte structure 1:word structure	00:forbidden 01: send data from master 10: receive data from master 11: send and receive data	0000: 1 byte or word 0001: 2 bytes or words 0001: 2 bytes or words 0010: 3 bytes or words 0011: 4 bytes or words 0100: 5 bytes or words 1111: 16 bytes or words

Example: 16 words DI at consistent length as send data to the master>11011111 = 223 dec.

16 words DO at consistent length as receive data from the master>11111111 = 239 dec.

Process the data in the AG95U-DP/Slave for sending and receiving.

For sending to the DP master and receiving from the DP master there is an extended I/O area available on the PLC linked to the areas of the process images. The extended I/O area for inputs (EPI) occupies addresses 128.0 ... 255.7 and the extended I/O area for outputs (EPO) occupies addresses 128.0 ... 255.7.

The extended I/O areas are simply for storing the DP data S5-95U. In the table you will find the precise structure of the extended I/O areas in the S5-95U. In the EPI you can read, in the EPO you can write.

EB32 from slave--> DB10.DBB0 in the master MB 10 from master---> AB32 in the slave

- Configuring the AG95U-DP slave as such in the S7 tool HWCONFIG:

Dialog box "DP slave properties" with module identification 223 as result for entering in the DB1 of the AG95U-DP slave:

I/O type: INPUT

Input>Addr: 128 Length: 16 Unit: WORDS Consistency: total length

Dialog box "DP slave properties" with module identification 239 as result for entering in the DB1 of the AG95U-DP slave:

I/O type: OUTPUT

Output >Addr: 128 Length:16 Unit: WORDS Consistency: total length

- Memory area of the extended I/O area in the AG95U-DP SLAVE:

SEND_BOX RECEIVE BOX

EPO EPI

128.0 5780 Hex 128.0 5700 Hex to 159.7 579F Hex 159.7 571F Hex

- S5 program in the AG95U-DP SLAVE:

L PY 128 from the master

T AB32 to the slave

L EB 32 from the slave

T PY 128 to the master

c- Quality Control File: Included with the files for each driver is a file of the same name with the ".sp" extension. This file contains the model and serial numbers of all hardware and software used for testing. This file also contains the setup parameters used for testing.

d-Scan Time: The key to getting the driver to function properly is to get the I/O updated and code executed at constant time intervals.

The S5 executes the driver code in a time-controlled program "OB 13". The time interval is contained in DB1 at block ID "TFB:". The time entered is in milliseconds; it must be in increments of 10 milliseconds and can contain a value between 20 to 250 msec.

The time-controlled program uses its own PII (Processor Input Image) for the IO. The inputs are automatically updated at the beginning of the time-controlled program and outputs are written at the end. No filters can be set on the inputs. They have a fixed filter between 2.0 and 2.5 msec. Some instability of the driver was noted when the time controlled program interval was set to 10 msec. This is probably do to the filter time being so large compared to the scan time. At 20 msec no instability was observed.

e- Timing Parameters Exceptions: The MSP_IO_M driver is the largest. This driver effectively has 4 channels. At 10 msec this driver was too long to fit in the execution time of the time-controlled program and caused the S5 to crash. For any of the multiplexed drivers that are expanded more than 4 channels the scan time should be watched closely and may have to be increased.

f- Transferring Program: When transferring the driver program to the PLC or another program, make sure to include the following blocks: OB13, FB101, FB102, and DB2. Also, the time-controlled program interval time must also be changed in DB1. This can either be done manually or by downloading D1 from the driver program.

g- Data Blocks: If you need to use a different data block other than DB2 it can be changed easily at the being of OB13. After designating the data block to be used make sure to also create and download a block of sufficient size to handle the number of channels used.

By default we set the Delta Counter Preset to be 16. If creating a new block or adding output channels be sure to set the Delta Counter Preset. The driver program controls all other values in the driver table (with the exception of the MSP Output Data).

h-Function Blocks: Most of the driver program for the S5 family is in two function blocks. FB101 is the function block for the inputs and FB102 is the function block for the outputs. Since function blocks allow parameters to be passed back and forth this makes addition of channel to the program very easy.

Function blocks in many ways have there own instruction set and can not use ladder logic very well. The drivers function blocks are done exclusively in STL and are not displayable as ladder logic or control system flowcharts. Even for those comfortable with STL they will find the logic has been refined and optimized to a very high degree. In some ways this may make the driver programs harder to read but it is felt that efficient use of memory and execution time are the most important factors.

i- Programming Methods: One of the key programming methods is the use memory locations that are accessed both as registers and bits. We used the DW memory for this purpose. In several cases counters are programmed by using a shift register. A seed is planted in bit zero of the shift register word. To increment the counter by one the shift resistor is shifted one bit. This allows the current value of counter to be checked by testing a single bit in ladder, which is much more efficient than a whole register compare. These programs have been extremely optimized for both minimum scan time and memory usage.

j- Variables Display: To aid in trouble shooting several variable block displays are included in the program. Under the "Test" pull-sdown menu select either "Status Variable" or "Force Variables". After pressing F1 for "Fetch" the following Variable Blocks are available:

13 Data for available channels

100Driver scratch pad block DW0 to DW7

101 Input Block for Channel 1

102Output Block for Channel 1

See the manuals for details on using these displays.

k-Documentation: With each driver are two files that provide documentation of the program. MSP???LS.ini is a text and MSP???LS.doc is a Microsoft Word file. Each file contains listings of the OB1, OB13, and the Memory Assignment List. FB101 and FB102 are included as required by the driver. The MSP???LS.doc is been formatted to provide the much more readable listing.

l- High Speed Counter (HSC): The HSC driver uses less ladder logic memory. Depending on the value transmitted it may be faster or much slower than the Delta protocol. Unlike the Delta protocol the update time for the HSC protocol is not deterministic.

The HSC driver uses hardware counters that have a maximum response of 2.5 KHz or greater. On the S5-90U the hardware HSC is limited to 1kHz and is not usable by the MSP HSC protocol. The S5-95U has two (2) input channels but only one is usable for the HSC protocol. Channel A which has a max limit of 5 kHz is usable. Channel B which has a max limit of 2kHz is not usable. All frequency limits must be reduced if the PLC is connected to a programmer, operator panel, or SINLEC L1 network. See the manuals for additional information on the hardware counter. Other models will require a separate counter module. Even though this mode of operation may be possible the cost of the special module may make the use of this method uneconomical.

The scan time on the MSP must be set greater than the maximum scan time of the PLC. We set the MSP to a scan of 10 msec for test purposes. This allows some room for the user to add their program.

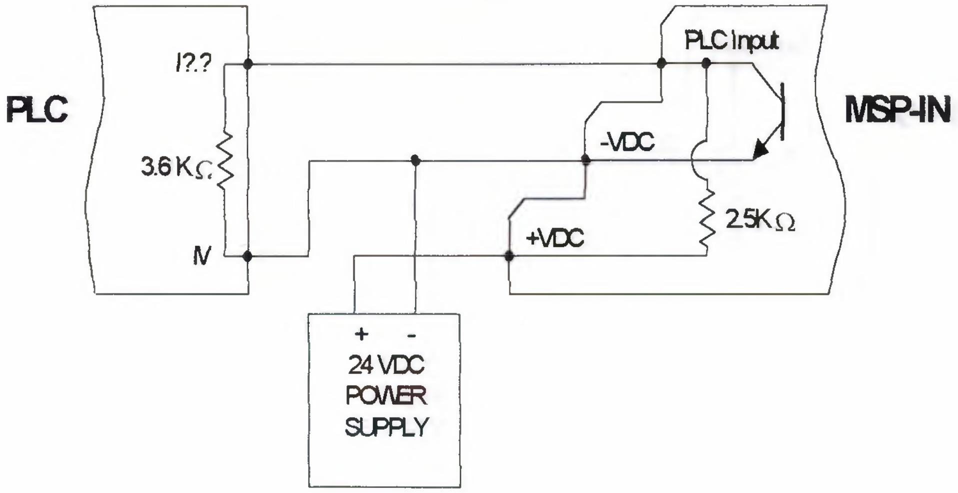
In order to send a value of zero (0) or negative values an offset is added to the pulse count before transmission. The driver then subtracts this offset after counting the received pulses. Note the subtraction that occurs in the second rung. The value of the offset varies depending on the MSP model and scale factor in order to keep the offset to a minimum. The following table shows the offset for the different ranges. The constant in the second rung must be changed to match the configuration of the MSP.

MSP MODEL	SCALE FACTOR	OFFSET
MSP-RTD	X1	50
MSP-RTD	X10	500
MSP-TC	X1	50
MSP-TC	X10	500
All other models		1

m- Sink Inputs: Almost all inputs on the S5 series are sink. Each input is basically a resistor with one end of each resistor tied to individual terminals. The other ends are tied together to a terminal marked "M". This pull down resistance sinks voltage and current to common VDC.

The output on the MSP is designed to be used as either sink or source. The MSP contains a sinking gate and pull-up resistor. Sink inputs can be wired directly to the MSP.

Make sure that the PLC input module and MSP input get their 24 VDC from the same source. Following is a schematic of a typical input. "I?.?" may be any valid Input. In the single channel drivers for the S5-95U used for testing the input is I32.0 but may be changed if desired. In the multiplexed drivers for the S5-95U Channel 1 and 1 are I32.0 and I32.1 respectively but may be changed if desired.



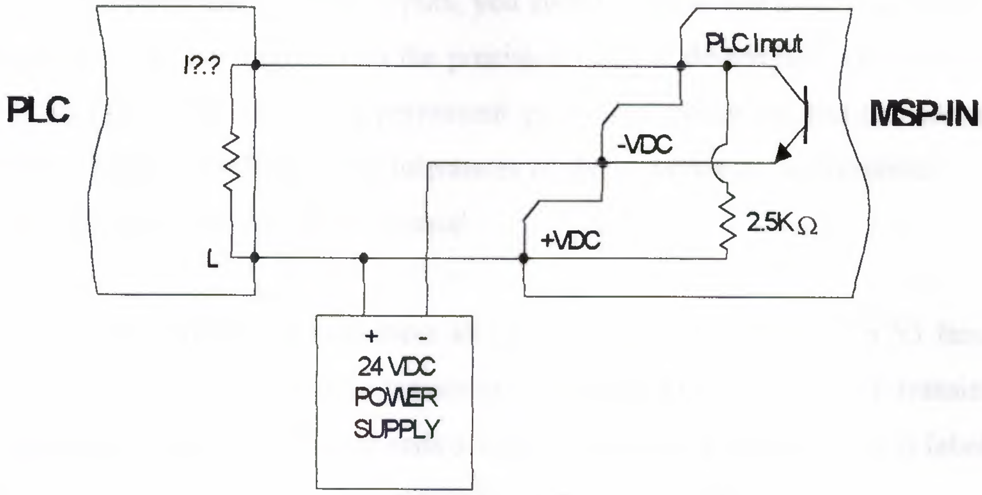
n-Source Inputs: Source DC inputs are rare in the S5 family but are available. Each input is basically a resistor with one end of each resistor tied to individual terminals. The other ends are tied together to a terminal that is marked "L". This pull up resistance sources voltage and current from +VDC.

The output on the MSP is designed to be used as either sink or source. The MSP contains a sinking gate and pull-up resistor. Source inputs can be wired directly to the MSP.

Using the input as source would increase the level of the signal and make it more immune to noise but it would double the current required when the signal is pulled to zero volts.

Make sure that the PLC input module and MSP input get their 24 VDC from the same source. Following is a schematic of a typical input. "I?.?" may be any valid input.

Note: The true/false sense for a sourcing input is opposite that of a sinking input. In ladder logic the hardware input into the driver must be negated to provide the proper logic sense.



1- Relay Outputs: Relay outputs can be used as either sink or source. We recommend using them only for source since it eliminates the need of a pull up resistor. See the next note on Sourcing outputs for a schematic and more details. If using mechanical relay outputs you will have to slow the output driver down in order to debug it and not destroy the relay. We suggest using a scan time of 50 msec for the MSP analog output modules.

2-Output on Demand: In order to conserve the life of the relay output we suggest you add some extra logic that allows control of when the output is transmitted. If required please review the Omron or GE drives for the method.

3- Input/Output Scan Time Ratio: For those drivers using both inputs and relay outputs we suggest using logic that runs the output a factor of 5 times or more slower than the input. This allows the input to run at a scan time of 10 msec and the output to run a scan time of 50 msec. This is necessary only when relay outputs must be used. If required please review the Omron or GE drives for the method.

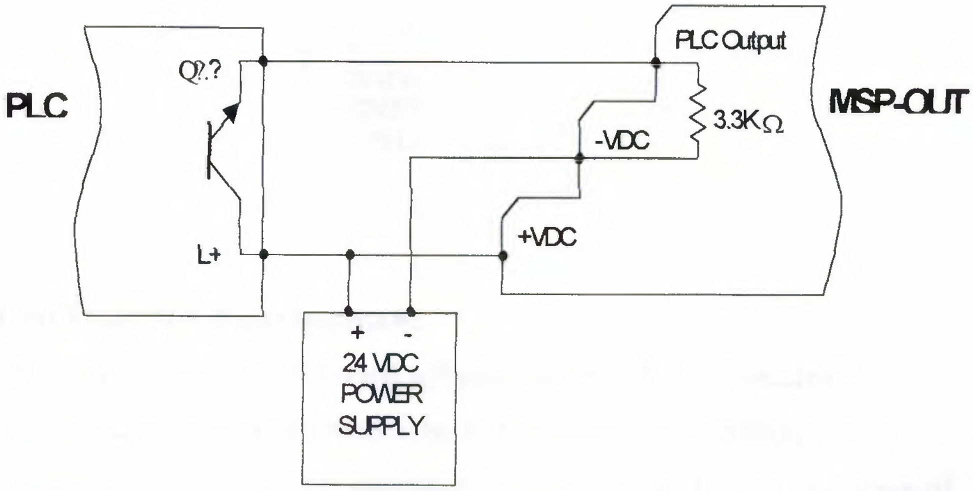
The analog inputs of the AG 95U can also be used as digital inputs. The usual range when used as analog inputs is -10 V to +10 V. The possible switching is from -10 V to +30 V. This range is used when implementing the analog inputs as digital inputs (usually 24V).

If, after using them as digital inputs, you switch back to use as analog inputs, it might happen that you no longer attain the precision that the device had upon delivery. The reason for this is that there is a permanent protective circuit beyond the nominal range, which can have an effect on the tolerances of the components implemented. You can find more details in the AG 90/95 manual

o- **Source Outputs:** All most all solid state DC outputs on the S5 family are source. Each output is basically a transistor or gate with one end of each transistor tied to individual terminals. The other ends are tied together to a terminal that is labeled "L+". This pull up contact sources voltage and current from +VDC.

The input on the MSP is sink; it is basically a resistor tied to -VDC or common. Source outputs can be wired directly to the MSP.

Make sure that the PLC output module and MSP output get their 24 VDC from the same source. Following is a schematic of a typical output. "Q?.?" may be any valid output. In the single channel drivers the output is Q32.0 but may be changed if desired. In the multiplexed drivers Channel 1 and 2 are Q32.0 and Q32.1 respectively but may be changed if desired.

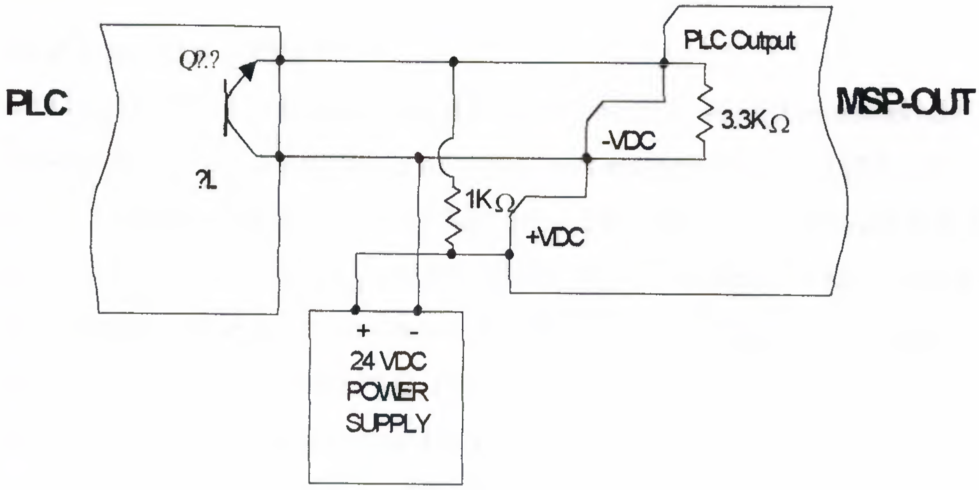


Sink Outputs: Solid state sinking DC outputs are rare in the S5 family but are available. Each output would basically be a contact with one end of each contact tied to individual terminals. The other ends are tied together to a terminal that is usually labeled "L+". This pull down contact sinks voltage and current to COM VDC.

The input on the MSP is sink; it is basically a resistor tied to -VDC or common. Since both PLC output and MSP are sink a pull up resistor must be used. Sink outputs can be wired directly to the MSP with a pull-up resistor.

Make sure that the PLC output module and MSP output get their 24 VDC from the same source. Following is a schematic of a typical output. "Q?.?" may be any valid output.

Note: The true/false sense for sinking outputs is opposite that of the sourcing outputs. In ladder logic the hardware output from the driver must be negated before going to the hardware output to provide the proper logic sense.



Real Time PLC Fundamentals:

The Real Time PLC is a straightforward software solution that is executed in a Windows environment as an interpreter. The PLC program is executed in a non compiled form, just the same as it is executed in a hardware PLC. The advantage of executing a PLC program with an interpreter is that the PLC status can be displayed in real time without any recompile activities. Diagnosis and testing of a PLC program is much easier and the instruction to be executed can be monitored in real time.

Real Time PLC Operating Systems:

To meet the demands and requirements of our customers and provide flexibility, different hardware and software platforms are available to execute the Real Time PLC. The operating systems Windows 3.1x/95 or Windows NT 4.x can be used. The user also has the ability to install and activate the Real Time PLC on an additional processor board which can be plugged into the motherboard of the PC.

Real Time PLC Compatibility:

As a result of a variety of performance requirements and the need to transfer existing PLC programs, without any modifications into a modern PLC environment, two versions of the Real Time PLC's are available. The Real Time PLC version PLC43 is compatible to the Simatic PLC CPU943. The version PLC45 is compatible to the Simatic PLC CPU945.

Real Time PLC I /O's

Standard I/O boards may be accessed with the Real Time PLC as well as numerous intelligent hardware boards are available to control bus systems widely used by the industries (e.g. CAN-Bus, Inter Bus, Profi Bus, etc.). There is even a hardware board that will allow you to connect an original SIEMENS S5-115U extension rack, with all possible I/O boards, available. Drivers have been developed to connect bus system interface boards with the Real Time PLC. Drivers for additional boards are easily programmed with Function Blocks using the standard STEP® 5 PLC programming language plus some additional special instructions supplied with the Real Time PLC. Very often only a handful of instructions are needed to realize such a driver.

PLC43 / PLC45 Running under Windows NT

On the Windows NT platform the Real Time PLC is executed as a task with an extremely high priority. This guarantees that the real time capability of the PLC43 / PLC45 is maintained.

Single Processor Systems

In the Windows NT environment with a single processor motherboard, the Real Time PLC simultaneously shares the CPU power with the other applications to be executed . Fifty percent (50%) of the CPU time (time slice) is assigned to the Real Time PLC and the other 50% is used by the other applications. The Real Time PLC is called within a preset time-frame of 2ms.

Double Processor Systems

In the Windows NT environment with a dual processor system one CPU is totally allocated to the Real Time PLC. The PLC cycle time depends directly on the execution time of the OB1. If OB1 has a short execution time, the number of cycles per time period is high.

PLC43 / PLC45 to be Executed on an Additional Processor Board

It is also possible to execute the Real Time PLC software on an additional processor board completely independent from the PC hardware. The additional processor board may be powered with its own power supply. Such a setup has the advantage that a PC failure does not impact the Real Time PLC. Even when the PC is turned off, the PLC program execution is not interrupted. Well known manufacturers offer processor board with integrated bus interfaces (e.g. Can Bus, Inter Bus, Profi Bus).

PLC43 / PLC45 Running under Windows 3.1x / 95

The Real Time PLC, to be executed under Windows 3.1x / 95, is developed as a device driver. The power of the CPU of a Personal Computer (PC) is split between Windows 3.1x / 95 and its application and the execution of the Real Time PLC. The portions may be set individually. For instance, 50% of the CPU power may be used by Windows and its applications, the other 50% is used by the Real Time PLC. The fixed time slice of the CPU power gives the PLC real time performance. Hotkeys can be defined. Hotkeys are hardware devices (push-button, key, switch, etc.) controlling a special task. It is possible to simulate several keystrokes with the push of one key and to inhibit the function of one or several keystrokes (e.g. Ctrl-Alt-Delete). The Real Time PLC has direct access to the PC monitor to display information.

Monitoring, Operating, and Controlling Simultaneously, from one PC

The Real Time PLC integrated in a PC allows process monitoring, operation, and control from one hardware unit. The use of the Real Time PLC is especially useful if the process currently calls for a PC for data collection, process visualization, programming, or any other reason. Such a solution reduces the required hardware by a PLC-CPU and the corresponding communication processor

Real Time PLC Access via DDE

Like *S5 for Windows* the Real Time PLC also has a Hotlink DDE interface with multi-channel capability. Windows applications can directly access the Real Time PLC data via this interface without *S5 for Windows.* It is even possible to communicate with the Real Time PLC over networks using the Windows function Net-DDE. Nearly all the visualization programs support the DDE interface. Using DDE no special drivers are required.

Real Time PLC Access via DLL

For fast data transfer between the Real Time PLC and other Windows applications, a DLL is available. Examples for Visual Basic, Excel and C are supplied. Numerous visualization software suppliers have DLL's available for fast data transfer between the Real Time PLC and their visualization software.

Interrupt Control for Fast Reaction

The Real Time PLC is capable of reacting on external interrupts to control fast events. This feature is used in conjunction with the Organization Blocks OB2, OB3, and OB4.

Integrated Watchdog Function

Intelligent up-to-date Bus Control Boards today have integrated watchdog functions. These watchdog functions guarantee that the inputs and outputs of the Real Time PLC, in case of a hardware failure of the PC, are handled in the same way as a hardware PLC CPU. If the PC fails, all the outputs of the Bus System are reset.

Direct Memory Access

Intelligent cards can use special instructions (DO RS 22, DO RS 23) to read and write to memory locations (e.g. dual port memory).

Hardware Access

To connect the PC executing the Real Time PLC with the outside world, an interface board is required. Small applications may only require Multi-I/O boards. These boards can be accessed directly from the Real Time PLC without major effort. The Real Time PLC provides special instructions to initialize standard boards (port address) and to read inputs and write to outputs. It is also possible to connect the PC to an external Bus System. In addition to standard I/O boards, different bus systems can be accessed via the Real Time PLC. Numerous intelligent hardware boards are available to control bus systems widely used by the industries (e.g. CAN-Bus, Inter Bus, Profi Bus, etc.). To access bus interface boards drivers have been developed. Drivers for the following bus systems are available:

Inter Bus S (boards from Phoenix and Hilscher), Profi Bus DP (board from Hilscher), Simatic 115 U extension rack (board from IBH softec). It is also possible to write drivers using special instructions (DO RS 10 up to DO RS 31) provided by the Real Time PLC within Function Blocks (FB).

Technical Details Relating to the Real Time PLC

The Real Time PLC as explained above is available in two (2) configurations, the PLC43 and the PLC45. The technical data for both versions is shown in the following chart. The execution time of the PLC instruction or PLC program with the Real Time PLC depends on the power of the PC processor used. The following chart, therefore, references different processor types. The PLC 43 and the PLC 45 have an integrated PID control algorithm (OB 251). Further more, the PLC 45 can also be used in a four accumulator mode (CPU 928 compatible).

*) for one (1) binary instruction **) Time slice setting: 50% PLC-time, 50% Windows-time. 1024 mixed instruction set (50% binary, 50% digital)

Programming S5 Communication

There are three data handling blocks used by the H1CP1314 DDE server, the SYNCHRON, SEND and RECEIVE

Function blocks. The function block numbers differ from one Siemens controller family to the next. The following table shows which FBs belong to which PLC.

LC	S5-115U/H	S5-135U/155H	
DB	CPU: 941, 942,942R 943, 944,945	CPU: 922,928,948,948R	CPU: 946/47,946/47R
SEND	FB244	FB120	FB120
RECEIVE	FB245	FB121	FB121
SYNCHRON	FB249	FB125	FB125
SEND_ALL	FB244(ANR=0)	FB126	FB126
RECEIVE_ALL	FB245(ANR=0)	FB127	FB127

When the PLC starts up, the CP interface to be used must be synchronized by the SYNCHRON handling block. Since this applies to every type of start-up with the PLC, the required number of SYNCHRON blocks must be programmed in:

- OB20 for a cold restart
- OB21 for a manual warm restart

- OB22 for a warm restart following power outage

The following is an example of OB20, OB21 and OB22

OB20, OB21 and OB22

JU FB User typed in call to Function Block 249.

NAME: SYNCHRON

SSNR: KY 0,**n** n should match the setting of Base SSNR in CP Init configuration.

BLGR: KY 0,**m** m=0 use maximum size available.

PAFE: FY **x** Diagnostic error byte, x= any available flag byte.

Implementing SEND and RECEIVE Function Blocks in Organizational Block1.

OB1

JU FB 126

Name: SEND-A

SSNR: KY 0,**n** n should match the setting of Base SSNR in CP Init configuration.

A-NR: KY0,**e** Job number, e=0 Send all

ANZW: FW **x** Job status flag word, user selectable.

PAFE: FY **x** Selectable flag byte for error status and diagnostics.

:

:

JU FB 127

Name: REC-A

SSNR: KY 0,**n** n should match the setting of Base SSNR in CP Init configuration.

A-NR: KY0,**e** Job number, e=0 Receive all.

ANZW: FW **x** Job status flag word, user selectable.

PAFE: FY **x** Selectable flag byte for error status and diagnostics

Mapping INPUT and OUTPUT blocks to DATA blocks

Since H1CP1413 DDE server cannot read/write the input and output blocks from the PLC memory, you have to copy the input and output blocks to data blocks.

Create two data blocks (Ex. DB100 and DB101), call DB100 the input data block and DB101the output data block. PICS writes to the input data block and the PLC copies the data to the input block. The PLC copies from the output block to the output data block and PICS reads data from the output data block. Using the **L** and **T** operations to transfer DW(words in the input data block) to IW(words in the input block) and transfer QW(words in the output block) to DW(words in the output data block).

Program example:

```
C DB 100 C DB 101
L DW 0 L QW 0
T IW 0 T DW 0
L DW 2 L QW 2
T IW 2 T DW 2
L DW 4 L QW 4
T IW 4 TDW 4
```

Because the input and the output address list does not exist in the PLC, when you transfer data between the input block, output block and data block, it causes the PLC go to stop mode. To avoid this problem, program OB 23. You add block end operation BE in the OB 23. You also need to edit DX0 in DB screen forms for your PLC.

Note: The OB 23 function varies in different PLCs. Check the PLC Programmable controller manual for more information.

Siemens Memory Usage

S5-95U

When the PLC is started (the soft-PLC is loaded into the PC memory and execution begins), a PLC image is loaded into memory. I/O and data table memory is initialised.

When the PLC moves into the RUN state the all inputs are moved from the I/O table into the data table, the engine then begins solving the logic. When all the logic has been solved the outputs are copied from the data table to the I/O table. The logic solve cycle, or program scan then begins again.

The logic is solved starting with the function block referenced by the InitialFB register.

Housekeeping tasks a carried out in parallel with the logic solve cycle. These tasks involve timer update, system information upkeep and monitoring.

A separate process is concerned with communications and security. A TCP/IP communications server is provided to allow communication on several ports. Connections can be made to the server for downloading application software, editing and monitoring the operation of the software, data acquisition, alarm annunciation and other functions as appropriate.

The security module prevents unauthorised access to restricted functions as appropriate.

The PLC has a number of operating modes;

ode	Na me	Description
	FA ULT	The logic engine has faulted, a logic solve exception has occurred.
	ST ANDBY	The logic engine is in standby, logic is not scanned, I/O is not updated.
	TE ST	The logic engine is running, logic is solved, I/O is not updated.
	RU N	The logic engine is running, logic is solved, I/O is updated.
	DE BUG	As RUN, but debug info is printed to the debug stream.
	M ANUAL	The logic engine is in standby, logic is not scanned, I/O is updated. (This maintenance mode enables the outputs to be forced and the input table to be updated in order to test I/O)

PLC Modes

Variables and the Data Table

The data table will be implemented using a vector table of vector tables. Currently only one, two dimensional data table is supported. This table is local to the PLC but global to all function blocks, networks and statements contained by the PLC. Function Block scope variables are not currently supported.

A vector is a special case of an array, which resizes itself automatically as required.

Table entries will be implemented as objects.

Each data type will be stored in a different vector table. (see Addressing).

Data Types

The following data types will be supported. (Some data types will not be available until later releases of the package).

Derived data types, structures, and arrays are not currently supported.

Typ e	Description	its	R ange	IEC	ince
BO OL	Bit		-	2.3. 1	.0.1
SIN T	Short Integer		- 128 to 127	2.3. 1	uture
INT	Integer	6	- 32768 to 32767	2.3. 1	.0.1
DIN T	Double Integer	2	- 2^{31} to 2^{31}-1	2.3. 1	.0.1
LIN T	Long Integer	4	- 2^{63} to 2^{63}-1	2.3. 1	uture
RE AL	Real/ Floating Point	2	±1 $0^{\pm 38}$	2.3. 1	uture
LR EAL	Long Floating Point	4	±1 $0^{\pm 308}$	2.3. 1	uture

TIMER	Timer Data	2	0 to 2^{63}-1	extension$^{note 1}$	.0.1

Supported Data Types

Note 1: The timer data type is not strictly compliant with the IEC defined type and is more akin to the type that Allen Bradley programmers are familiar with.

Note 2: Exclusion from the above table does not mean that future implementations of the jPLC will not support a possible data type.

Note 3: Support for bit arrays is implicit using bit level addressing. This is more familiar than the 1131 defined means, is easier to implement and less complicated. 1131 style bit arrays may be implemented at a later date.

Addressing

Addressing of data table entries will be as per IEC 61131-3 (1993), with extensions.

There are two types of addressing, direct addressing and symbolic or tagname addressing.

Direct addressing, addresses program variables by their data table location. IEC 61131-3 denotes directly addressed variables by the use of the '%' identifier. The variable is represented by the '%' identifier, followed by a letter code, a modifier(optional), a table entry address or element number and a bit address i.e. *%TMe.b*.

Where, *T* = area designation, (IEC 1131-3, 2.4.1)

M = Variable modifier, (IEC 1131-3, 2.4.1)

e = the element number, (IEC 1131-3, 2.4.1)

and, *b* = bit number. (extension)

Supported type designations are as follows.

rea ode	Description	IEC	Since
	System memory area *element*	*extension*	0.0.1

	Input memory area element	2.4.1	0.0.1
	Output memory area element	2.4.1	0.0.1
	Flag/bit memory area element	2.4.1	0.0.1
	Timer memory area element	extension	0.0.1

Type/Memory Area Designations

Modifier	Description	IEC	Since
X	Bit	2.4.1	-
B	Byte	2.4.1	Future
W	Word	2.4.1	0.0.1
D	Double Word	2.4.1	0.0.1
L	Long Word	2.4.1	Future
AC	Accumulator (timers/counters)	extension	Future
PR	Preset (timers/counters)	extension	Future

Type Modifiers

Tag name addressing is the IEC/PLCOpen preferred method of addressing. Here the variables are addressed in the same way as High Level programming languages, by the tag name identifier. Tag names can be assigned to a direct address in the variable declaration.

Currently only direct addressing will be supported. Tag name addressing, with auto/anonymous assignment will be supported in a future release.

Auto assignment assigns the next available direct address of the correct type to each variable as it is declared. Auto assignment can take place in either the program development software or on compilation.

Anonymous assignment is auto assignment which takes place only on compilation. In this case the assignment is hidden from the user and subsequent downloads may assign different memory to the same variables (though it is probable that if no changes have been made the assignment will be the same).

Indirect or Indexed addressing has been found to be useful in the past. If possible these types of addressing will be supported in the future.

Variable Declarations

Variables are declared in the source code using the IEC 1131-3 notation.

As with all IEC 1131 declarations, the declaration is surrounded by block delimiters, in this case VAR and END_VAR. For example,

```
VAR
max_speed : REAL;
mode : INT;
END_VAR
```

declares two variables, a real variable with the identifier max_speed, and an integer, mode.

Different types of declaration can be made. The following table shows the declaration key words for various variable groups.

Keyword	Description	IEC	Since
VAR	Declares one or more variables	2.4.3	Future
VAR_INPUT	Declares externally supplied variables.	2.4.3	Future
VAR_OUTPUT	Declare externally used variables.	2.4.3	Future
AT	Memory address assignment modifier.	2.4.3.1	Future

Variable Declaration Keywords

Variables can be assigned a memory location when declared. This is done using the AT keyword e.g.

```
VAR_INPUT
stop AT %I1.1 :BOOL;
END_VAR
```

declares the input, stop, at input address %I1.1.

Note: the bit address in the above example is an extension to 1131.

Variables can be declared at any point in the source file providing it is not within a code block declaration. Note that local (FB scope) variables are not supported.

I/O

Although the earliest issues of the PLC are not expected to interface directly to I/O, it is worth some discussion on the requirement and implementation of the I/O.

First a definition of I/O in terms of the jPLC is required. I intend to define I/O at this point as the *direct data interfacing with processes external to the current instance of the JPLC* . This may include the interfacing to real world I/O, other logic engines, I/O servers or HMI/SCADA applications.

It is intended that this interface will connect to the Shared Memory Model supported and proposed by the Linux-PLC project. It is also intended that in (the far) future the I/O interface will support a number of connection methods.

As I/O drivers deal directly with hardware, these drivers will not be written in Java and will be platform dependant. The intention is to define a standard interface, which will allow alternative drivers to be used.

The I/O space from the PLC's view will consist of two word tables, inputs and outputs.

The PLC will NOT be concerned with the mapping of I/O to physical devices.

The PLC will, probably, use Java Native method Invocation (JNI) to access an external driver to perform I/O update.

The actual implementation will depend on the limitations of JNI and any requirements issues. At its simplest the input update method passes the length of the input table, len, to the driver along with the PLC's desigantion. The driver should then return an array List] containing the input data. The output update method is passed the PLC designation, the output table as an array and the length of the array.

The driver uses the PLC designation to determine, from the I/O configuration, which inputs need to be returned and which outputs need to be set.

Forces:

Forces are provided as a maintenance feature. Forces are a part of the PLC function and can force an output or any number of outputs to a predetermined state. The I/O driver/server/engine does not need to know anything about this.

Inverts:

Inverts may be provided as a convenience. An inversion is a function of the I/O engine and can reverse the state of a Boolean I/O point. The inversion is transparent to the PLC, which does not need to know of its existence.

Logic Engine

The logic engine solves the logic programmed in a list of tokenised instructions or statements.

A statement is an object which consists of several fields, the fields of most importance to the solver are the label, operator (with its modifier(s)) and operand.

The label is use as a target for program flow control operators (jumps).

The operator contains a token which represents an IEC 1131-3(3.2) instruction list (IL) command along with its modifiers.

The operand is the data on which the operator performs its operation.

Execution of the operator will monitor and/or modify the contents of the result object. This object stores the result of previous operations for use with the current one.

Instruction Set

The following table shows the supported operators and their tokens (op code).

	O perator	od[4]	perand	Comments	I EC	ince
	L D		ny	Load the result register with the current operand	3 .2.2	.0.1
	S T		ny	Store the contents of the result register in the current operand.	3 .2.2	.0.1
	S		OOL	Set the operand to true (1)[2]	3 .2.2	.0.1
	R		OOL	Set the operand to false (0)[2]	3 .2.2	.0.1
	A ND		OOL	Boolean AND	3 .2.2	.0.1
	O R		OOL	Boolean OR	3 .2.2	.0.1
	X OR		OOL	Boolean Exclusive OR	3 .2.2	.0.1
	A DD		ny[1]	Add the contents of the operand to the contents of the result register and place the result in the result register.	3 .2.2	.0.1
	S			Subtraction (see	3	

	UB		ny[1]	Add)	.2.2	.0.1
0	M UL		ny[1]	Multiplication (see Add)	3 .2.2	.0.1
1	D IV		ny[1]	Division (See Add)	3 .2.2	.0.1
2	G T		ny[1]	Compare the contents of the result register with the contents of the operand and set the Boolean result if Greater Than.[3]	3 .2.2	.0.1
3	G E		ny[1]	Greater than or equal (see GT)	3 .2.2	.0.1
4	E Q		ny[1]	Equal (see GT)	3 .2.2	.0.1
5	N E		ny[1]	Not equal (see GT)	3 .2.2	.0.1
6	L E		ny[1]	Less than or equal (see GT)	3 .2.2	.0.1
7	L T		ny[1]	Less than (see GT)	3 .2.2	.0.1
8	J MP		abel	Jump to label	3 .2.2	.0.1
9	C AL		ame	Call function block	3 .2.2	.0.1
0	R ET			Return from function block	3 .2.2	.0.1
1	)			Evaluate deferred operation.	3 .2.2	.0.1
01	L DN		ny	Load the result register with the inverse of the current operand	3 .2.2	.0.1
02	S TN		ny	Store the inverse of the contents of the result register in the current operand.	3 .2.2	.0.1
	A			Boolean AND	3	

05	NDN		OOL	NOT	.2.2	.0.1
06	O RN		OOL	Boolean OR NOT	3 .2.2	.0.1
07	X ORN		OOL	Boolean Exclusive OR NOT	3 .2.2	.0.1
18	J MPN		abel	Jump to label[2]	3 .2.2	.0.1
19	C ALN		ame	Call function block[2]	3 .2.2	.0.1
20	R ETN			Return from function block[2]	3 .2.2	.0.1
28	J MPC		abel	Conditional Jump to label[2]	3 .2.2	.0.1
29	C ALC		ame	Conditional Call function block[2]	3 .2.2	.0.1
30	R ETC			Conditional Return from function block[2]	3 .2.2	.0.1

Instruction Set

note 1: These operators are overloaded. They act on and give a result of the same type as the operand.

note 2: These instructions are conditional on the state of the current Boolean result. The instructions S,R and those ending in C are conditional on a true (1) result. Those ending in N are conditional on a false result (0).

note 3: Conditional operators set the Boolean result on execution. Statements take the result register to be the left hand argument in an expression (i.e. result := result .op. operand), e.g. LD a, GT b stores a Boolean true if a>b, or a false if b>=a.

note 4:IEC 1131 defines three modifiers. N inverts the operand on an input instruction, the result on an output instruction and the condition on a program control instruction. C defines a program control instruction to be conditional on a true Boolean result (see note 2). The (modifier defers the current operation until the) operator. This implementation defines operators with N and C modifiers with their own op code.

Program Organisation

The application program for the PLC is organised in a hierarchical modular manner compatible with the program organisational units defined in IEC 1131.

The smallest unit of the program is the statement. The statements are executed in order one after the other. Program flow is from the first statement to the last unless otherwise defined by use of JMP, CAL or RET.

Statements can be grouped into Networks. The Networks are executed in order one after the other. Program flow is from the first Network to the last unless otherwise defined by use of JMP, CAL or RET.

Note: The Network is a kin to the Modicon and Omron objects of the same name, and the Siemens segment.

Statements and/or Networks can be grouped together in function blocks (FB). Function Blocks are only executed when called by the use of the CAL instruction in a statement.

Function blocks can be nested, and there is no predefined level of nesting.

Parameter passing is key to the use of function blocks. (future)

There is at least one function block within a PLC, this is the initial function block which the PLC calls on start up.

Program Declaration

Programs are declared in a text source file.

The initial function block can be declared in a number of ways.

The preferred method is to declare a PROGRAM. The program may have function block declarations embedded, or may call other function blocks declared externally but within the same source file (the use of multiple source files may be developed in future).

(Only one program declaration per PLC will be supported initially).

In the absence of a PROGRAM declaration, the compiler will accept function blocks with the following names as the initial function block, in order of preference;

- INIT
- MAIN

Declaration, as with variables, utilises a BLOCK/END_BLOCK structure, i.e.

```
BLOCK BLOCKNAME
some statements
...
...
...
END_BLOCK
```

So a program might be declared as follows,

```
PROGRAM MIXER
LD %S1.1
CALC STARTUP
CAL WEIGHING
CAL MIXING
FUNCTION_BLOCK WEIGHING
weigh some ingredients....
..
..
..
RET (* As the last statement in an FB RET is implicit *)
END_FUNCTION_BLOCK
END_PROGRAM
```

This might be followed in the source file by;

```
FUNCTION_BLOCK MIXING
mix some ingredients....
..
..
..
END_FUNCTION_BLOCK
```

An option is to store program comments and annotations with the compiled program in the PLC, this provides the ability of uploading fully documented source directly from the PLC. This does not affect the performance of the PLC, unless comments are displayed when monitoring, it does however affect the memory required by the application.

Comments can be entered into the source file either in

- IEC 1131-3 accepted format, i.e. (* ......comment ....*)
- or in C++, Java short format, i.e. //.....comment

In either case only single line comments can be stored in the PLC.

Compiler

The compiler is built in to the download mechanism within the programming port. It is intended that this compiler is pluggable, which would enable various compiler options (all having the same final output). Indeed the default compiler may enable various notations to be downloaded transparently.

The compiler will be flexible and will recognise several variations on a theme, the preferred notation will however be IEC1131-3 compatible.

The download compiler will of necessity, probably use a multi-pass mechanism. The first pass will set up variable and function block references, the second pass will 'populate' networks and statements. Additional passes may be necessary to resolve label references.

The compiler will have and incremental mechanism to allow online modifications to be made.

A decompiler function will be associated with the upload and monitor functions.

4.6. COMMUNICATIONS:

One or more TCP/IP servers will be provided to allow communications between the PLC and external applications. There will be at least two servers each with their own port number.

Port No.	Function	Client Access	Comments	Since
TBA	Programming	Single	This port is used to upload and down load programs[2], make online changes[2] and to monitor program operation. Also allows read/write access to the data table[2].	0.0.1[1]
TBA	Data Access	Multiple	This port allows read/write access to the data table[2]. It also allows the program to be monitored and/or uploaded.	

Instruction Set

note 1:May not be fully functional at issue.

note 2:Subject to access rights.

Additional communications paths will be added in future.

Additional ports that may be included are Error and Debugging ports.

The communications servers work in conjunction with a Security server, which ensures that only appropriate access is allowed (future).

4.7. SYSTEM FUNCTIONS:

Various systems and housekeeping tasks are undertaken. These tasks include timer update, watchdog monitoring and built-in data flag update. The system tasks include the update of the System memory area of the Data Table.

Timer Update

It is worth a brief discussion on timer update. The timer, as are all data types, is implemented as a class and is instantiated as an object. Timers are updated by timer events produced by special threads. These events cause the update of one or more lists of timers, each list contains timers which have been assigned a particular time base.

The initial idea was, on an update event, simply to increment an accumulator register, compare it with a preset register, setting status flags depending on the result. This would be simple to implement, however, it would not be particularly accurate as the event time would not be determinant.

An alternative implementation is to store the system time in the timer when the timer's enable flag goes true. The update event would then check to see if the difference between the current system time and the stored system time was greater than the preset time. This would not have the possible creeping error inherent in the first method, it would also be closer and more easily convertible to the IEC Timer data type. It does however have it's own performance issues and several compromises will have to be made in its implementation.

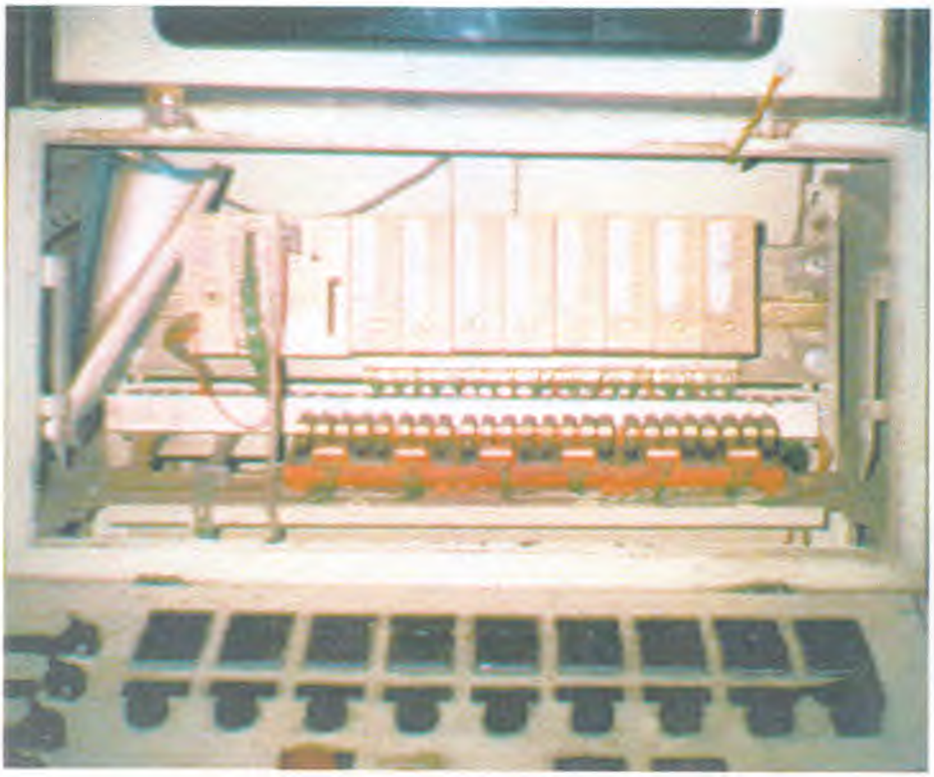
CONCLUSION

When developing this project we see that PLC the individual's life easier which it has gained our interest and notice.

With the information observed from our lecturer and our researchers for this topic PLC, is a convenent tool with a wide rage of useful ways to be used. Such examples can be mentiaoned severel machines can be used at the same time, easy adjustments from the PLC program can be meet within a few minutes by the keyboard, installed PLL programs can be controlled or checked before within the office and labaratory, even the PLC programs for firm can be meet at home. It is very protective and safe for the workers which they me protected from dager, communication programs of PLCs within each other or within operatus can happen with the PLC; the developed lantues have constructed the prodactivity, security, establishment security fast prodactivity, quality and we can see that PLC is a very cheap program that can be fundementelly used.

APPENDIX

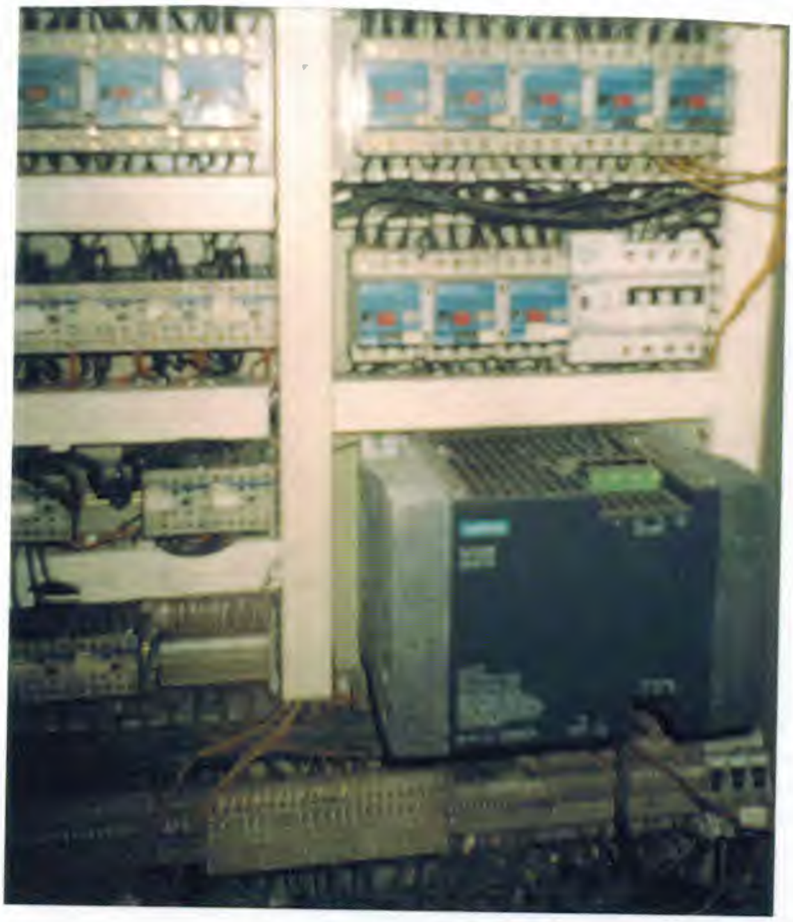
The front view of PLC machine



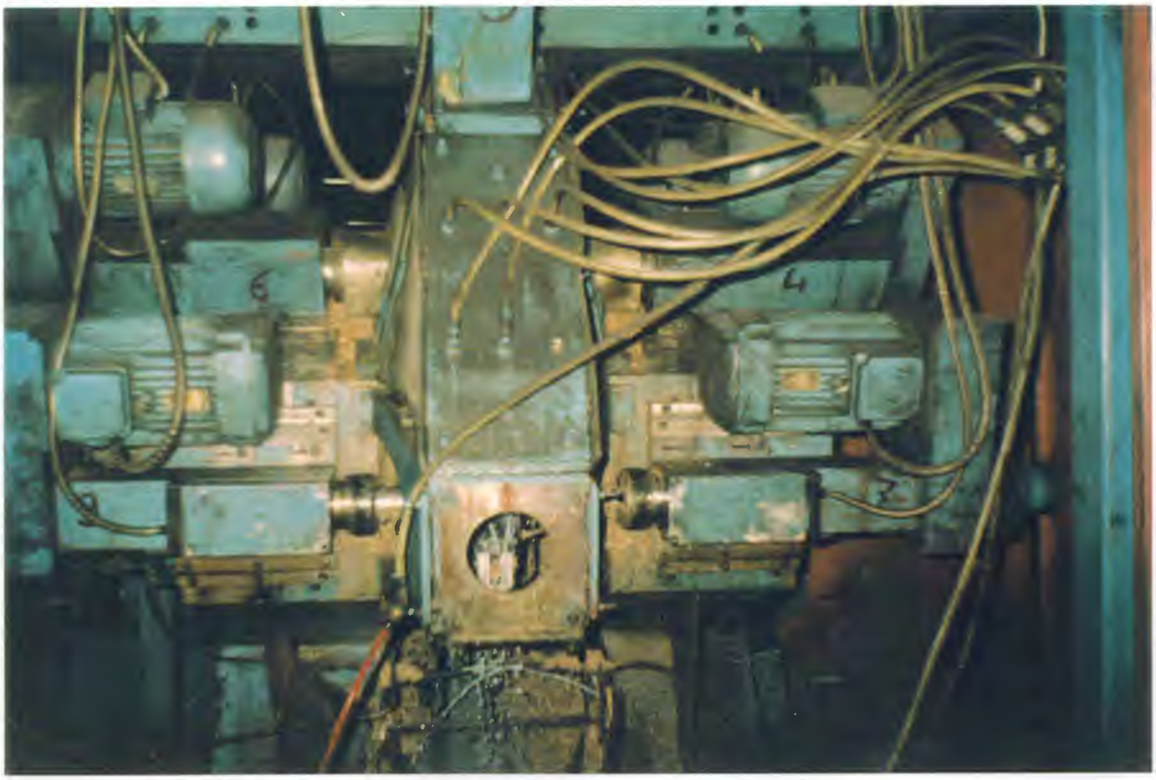
The view of Input and Output module

All of the PLC machine

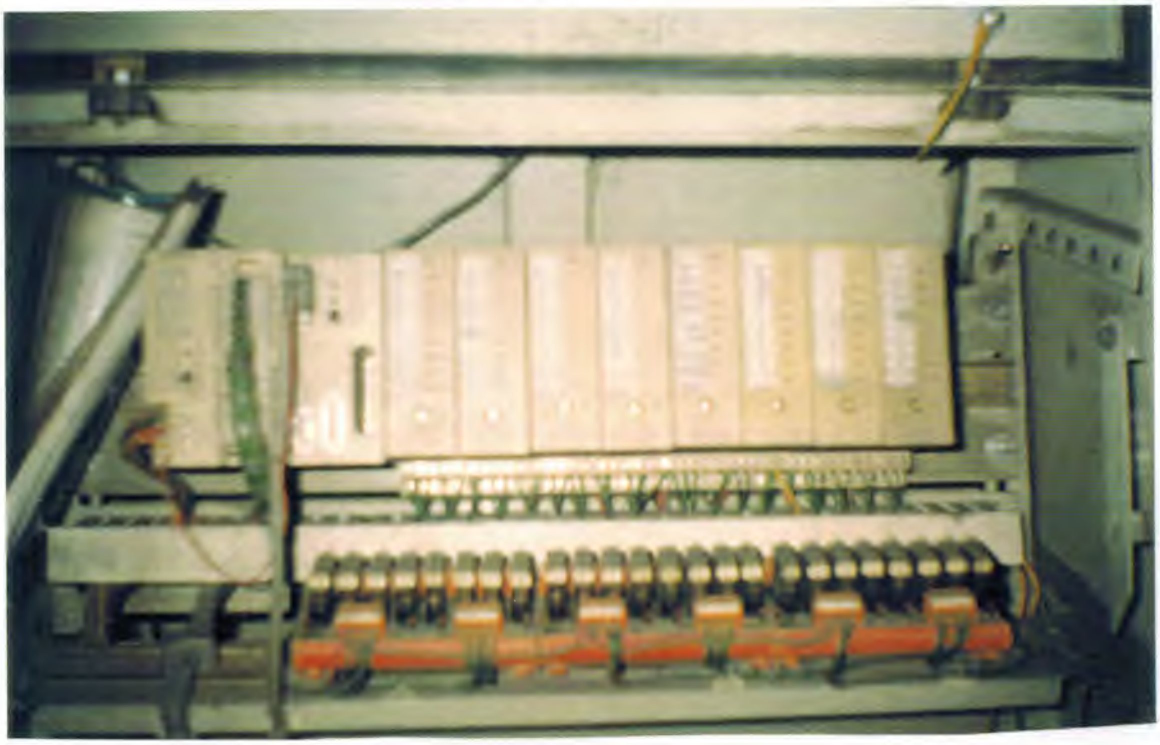
The view of CPU

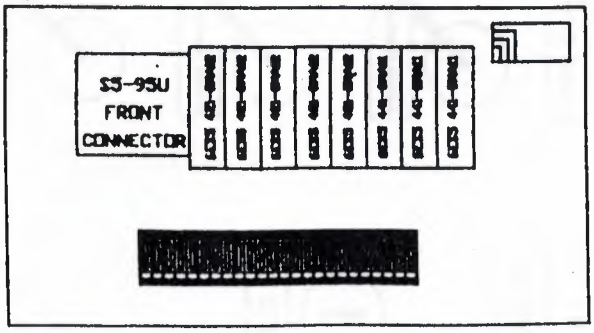


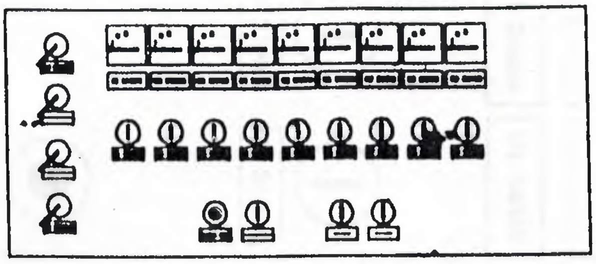
The links of PLC machine

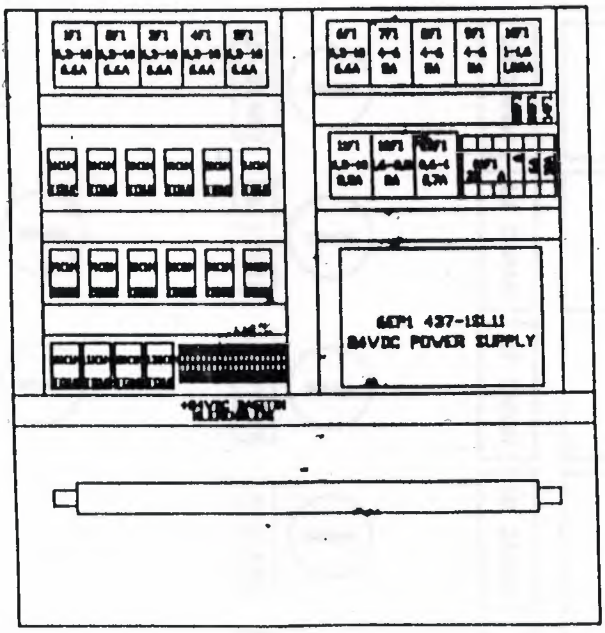


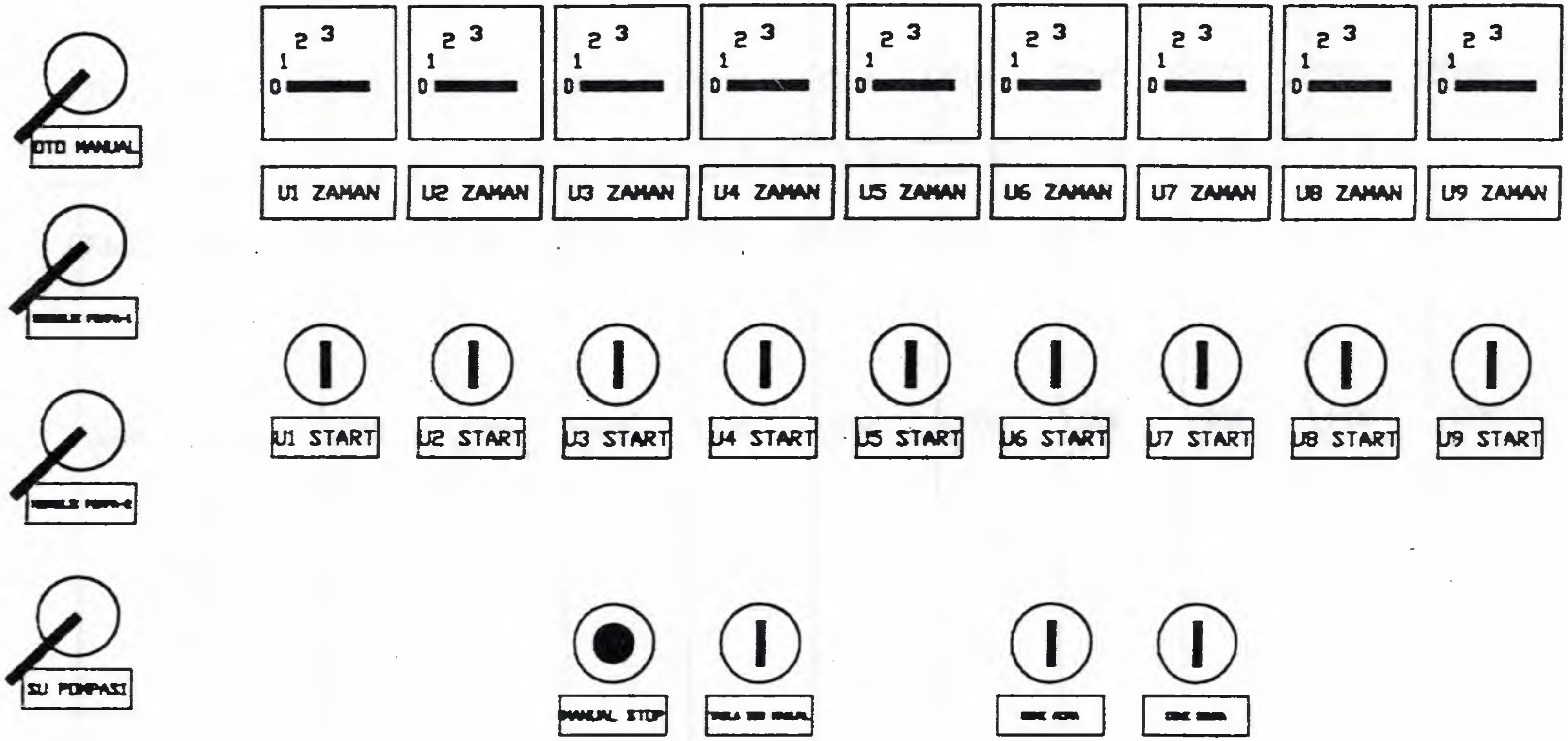
The motors which are controlled by PLC

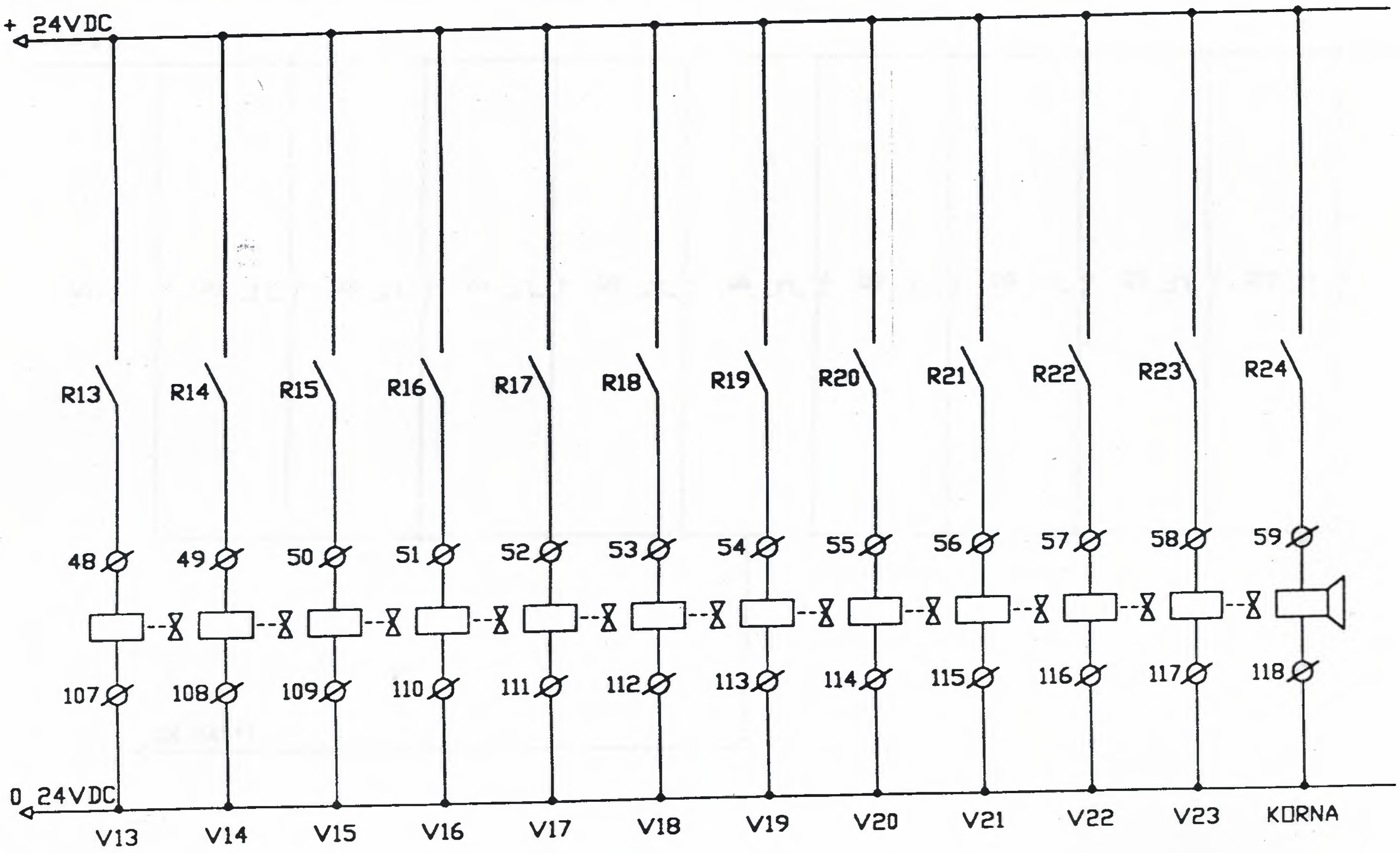



S5-95U
FRONT
CONNECTOR

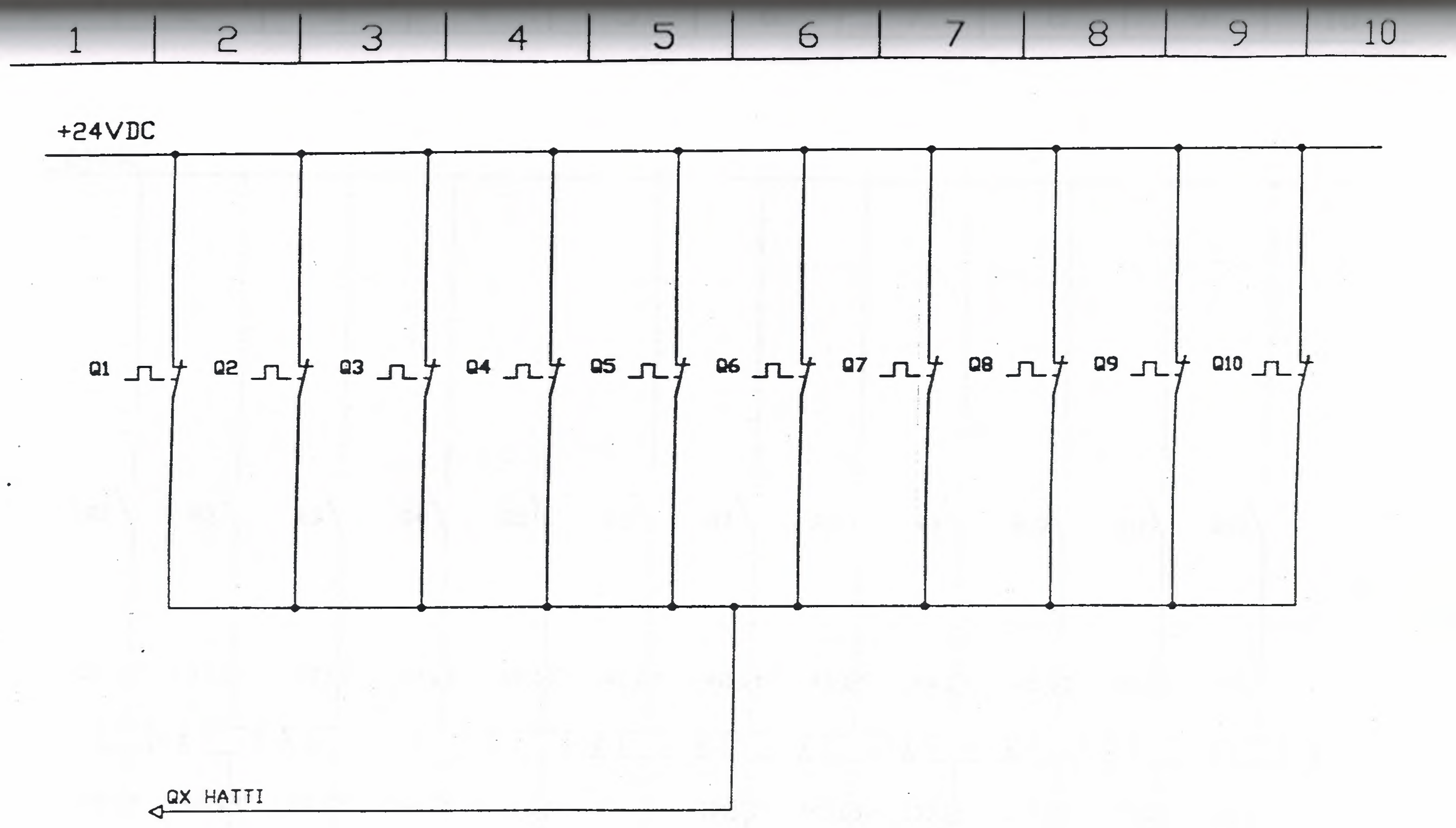



6EP1 437-1SL11
24VDC POWER SUPPLY

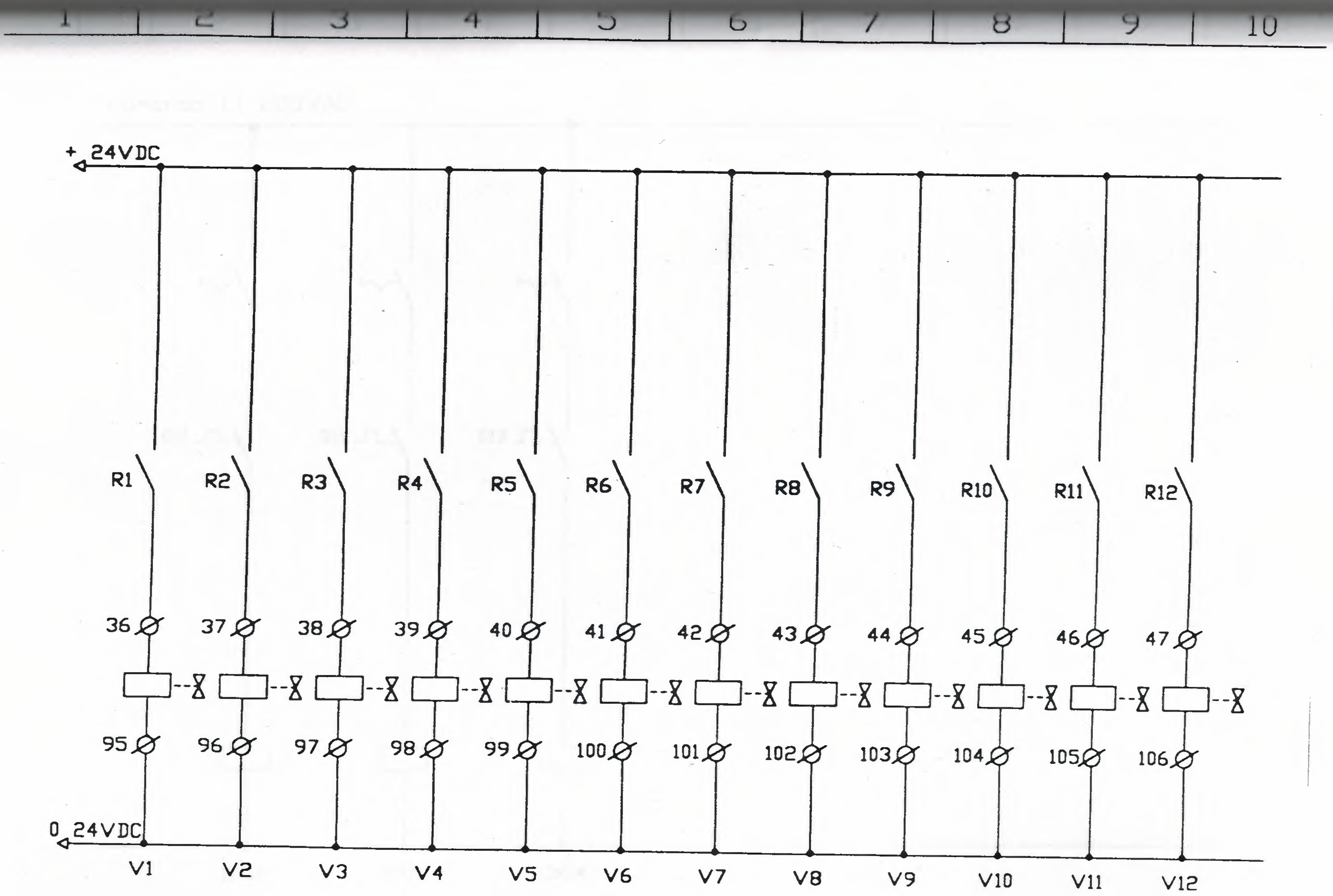

0 1 2 3
U1 ZAMAN
U2 ZAMAN
U3 ZAMAN
U4 ZAMAN
U5 ZAMAN
U6 ZAMAN
U7 ZAMAN
U8 ZAMAN
U9 ZAMAN
U1 START
U2 START
U3 START
U4 START
U5 START
U6 START
U7 START
U8 START
U9 START
OTO MANUAL
SU POMPASI
MANUAL STOP



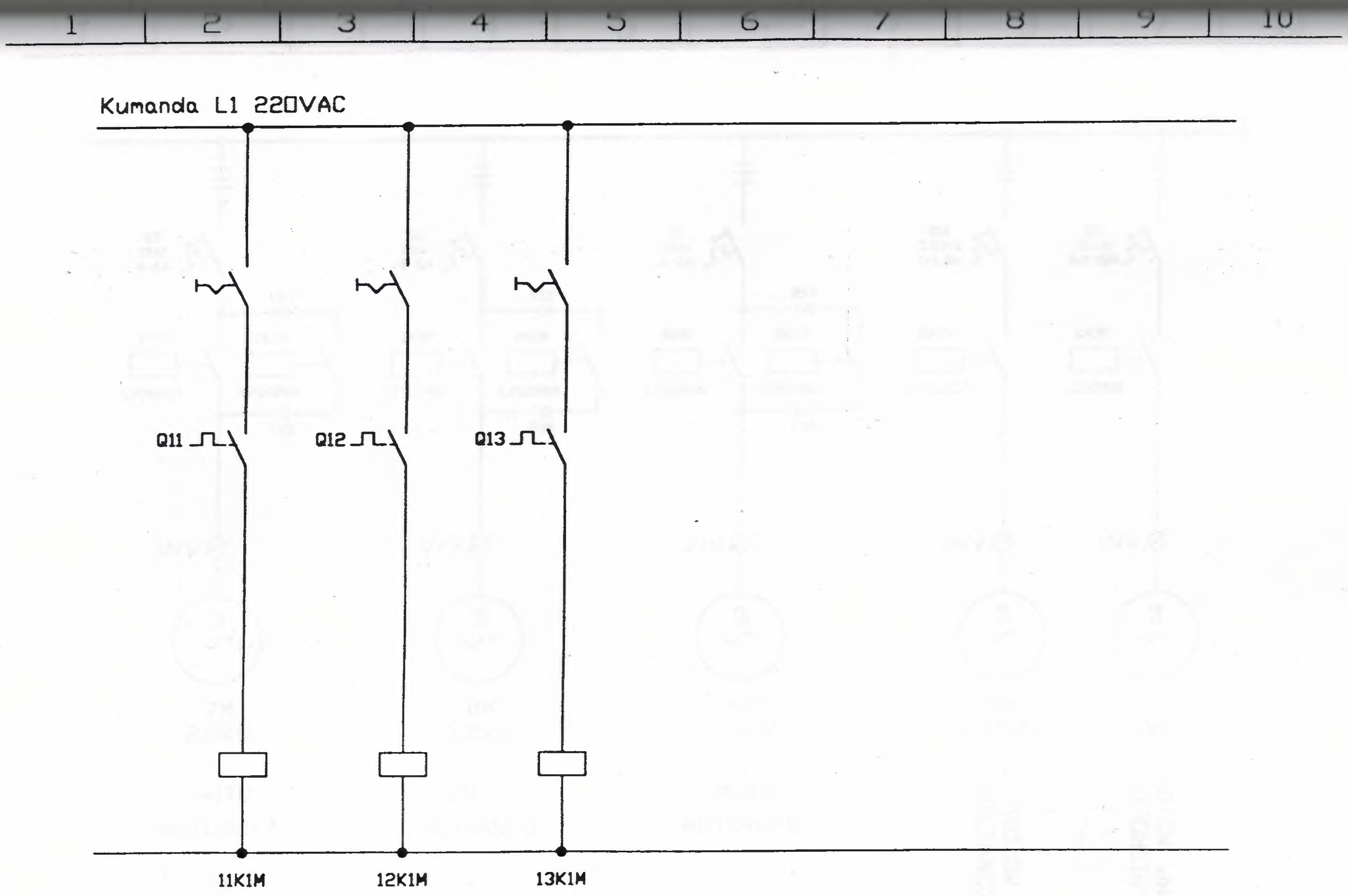
Yapan	Cizen	Kontrol	Tarih	IS NO: 53	Kod No
RIDVAN	SUAT	RIDVAN	25,04,94		95-53



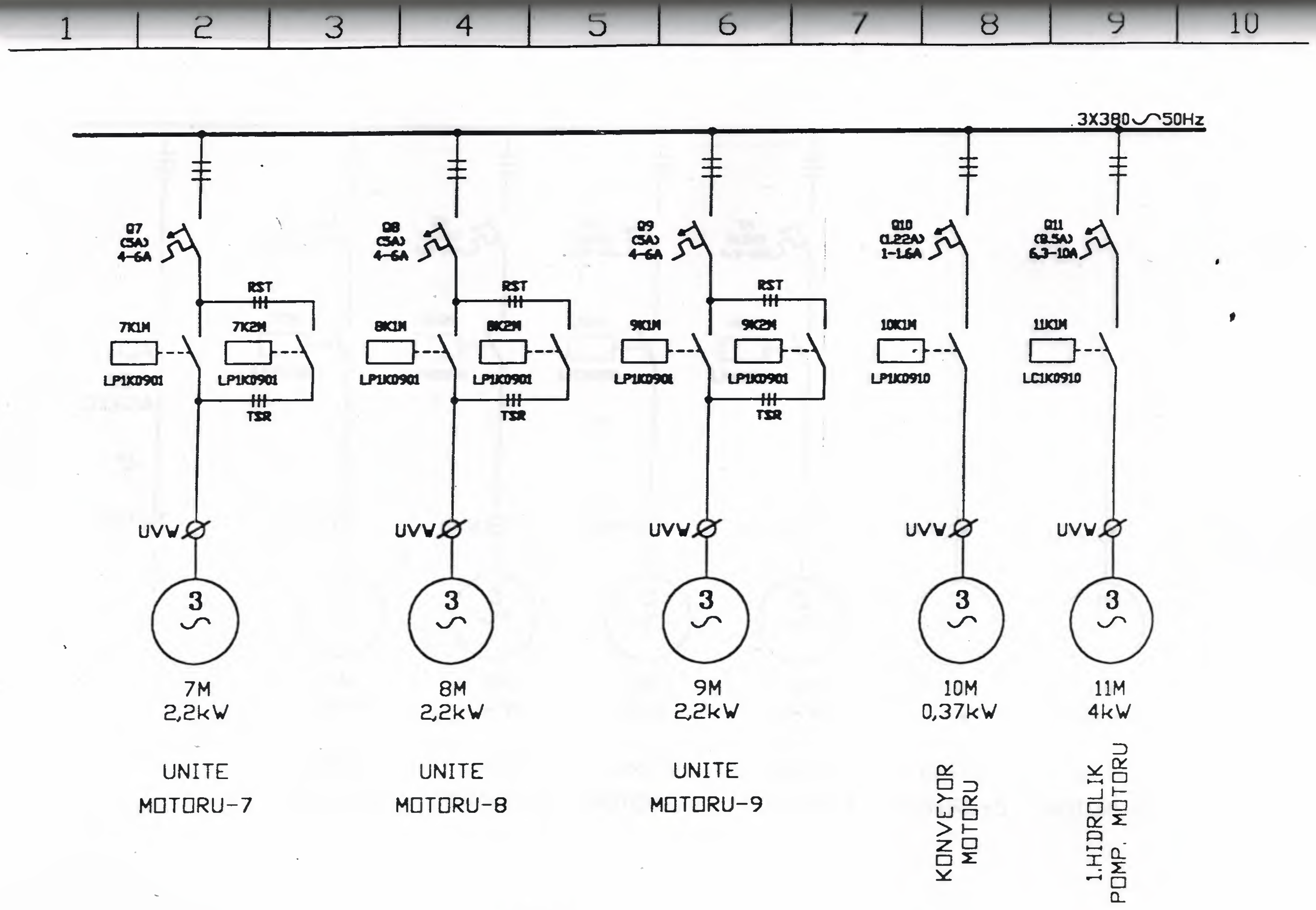
Yapan	Cizen.	Kontrol	Tarih	IS NO: 53	Kod No
RIDVAN	ERCU	RIDVAN	25,04,94		95-53



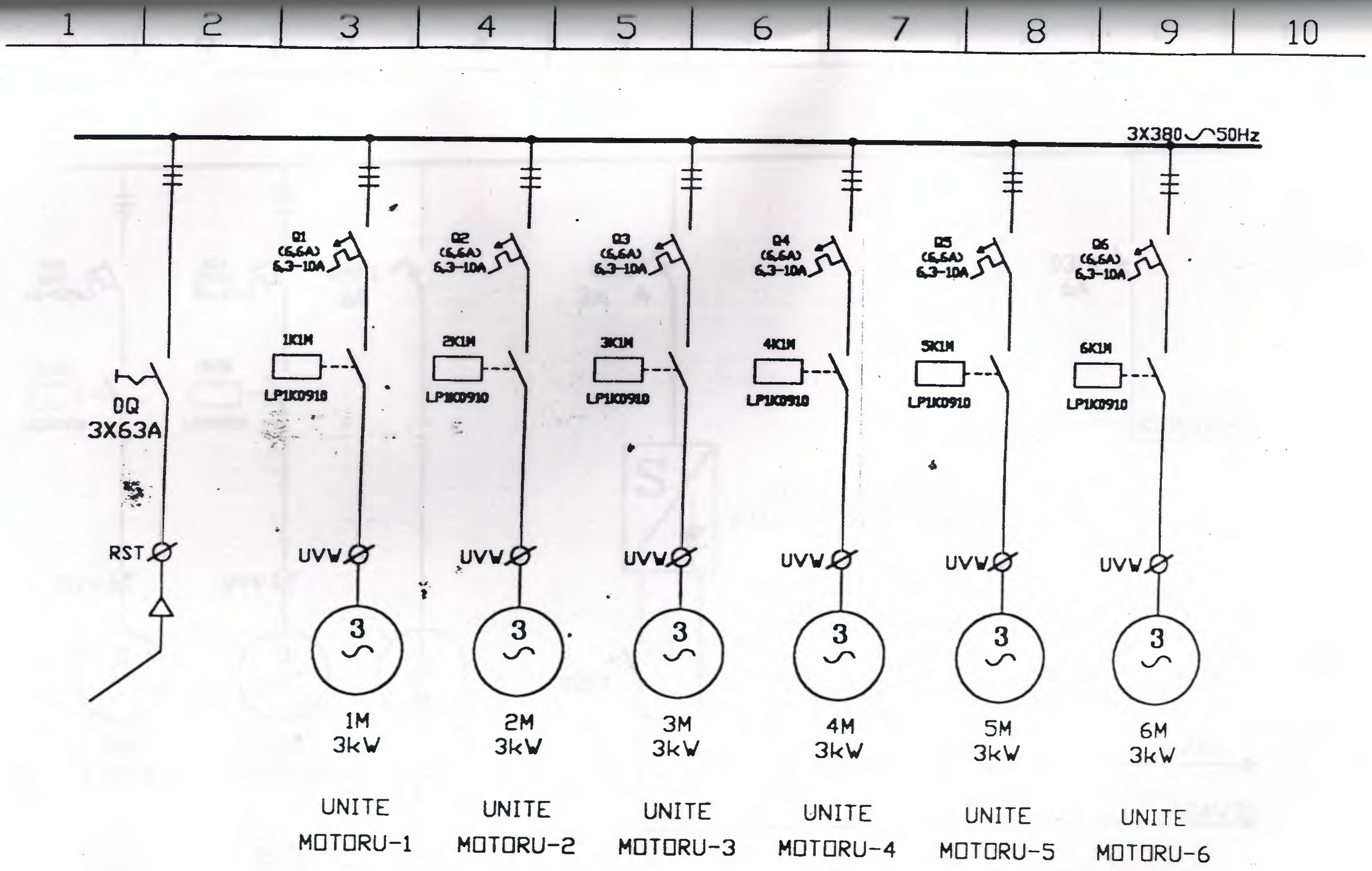
Yapan	Cizen	Kontrol	Tarih	IS NO: 53	Kod No
RIDVAN	SUAT	RIDVAN	25,04,94		95-53



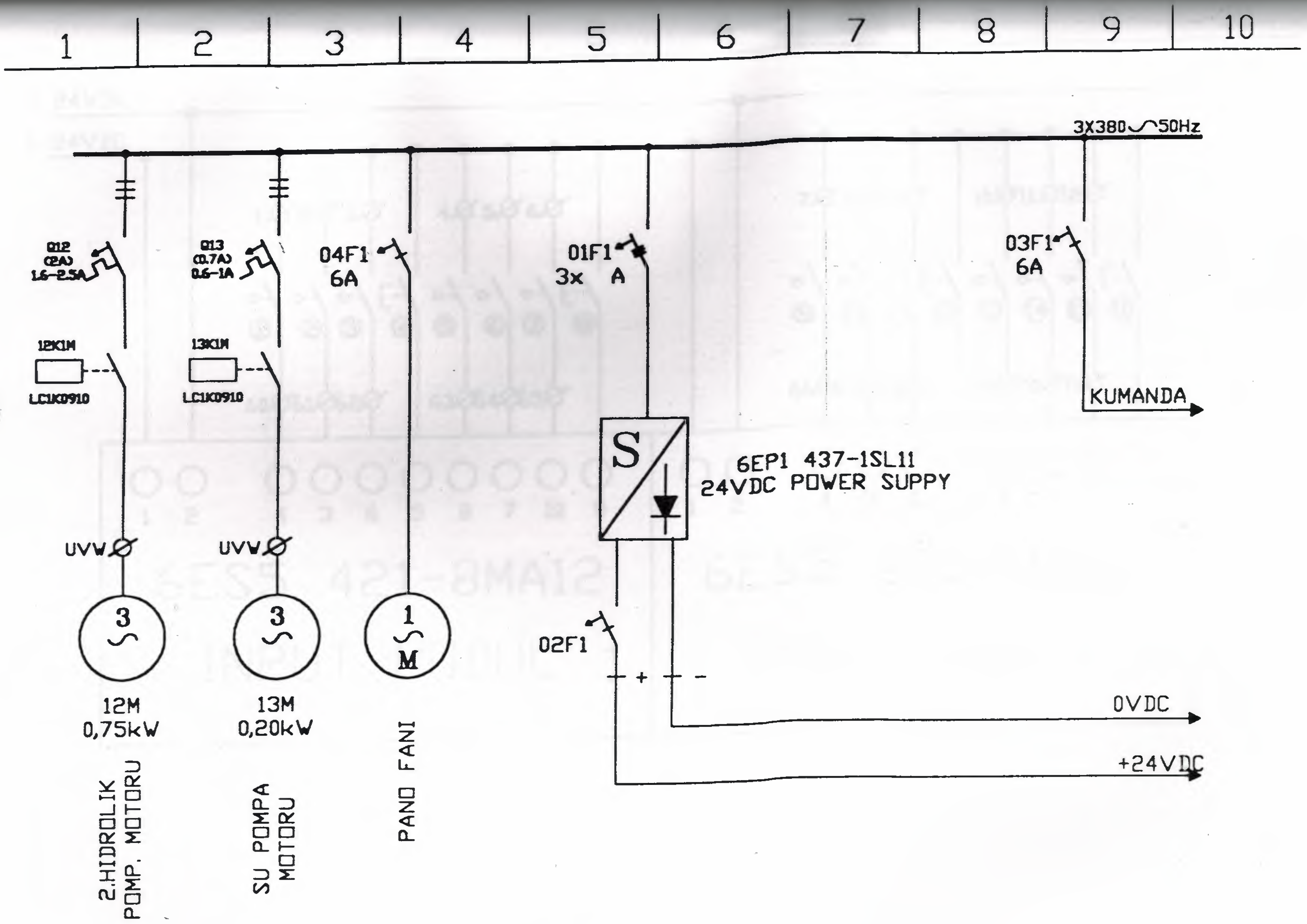
Yapan	Cizen	Kontrol	Tarih		IS N□: 53	Kod No
SUAT	SUAT	RIDVAN	24.02.95			95-53



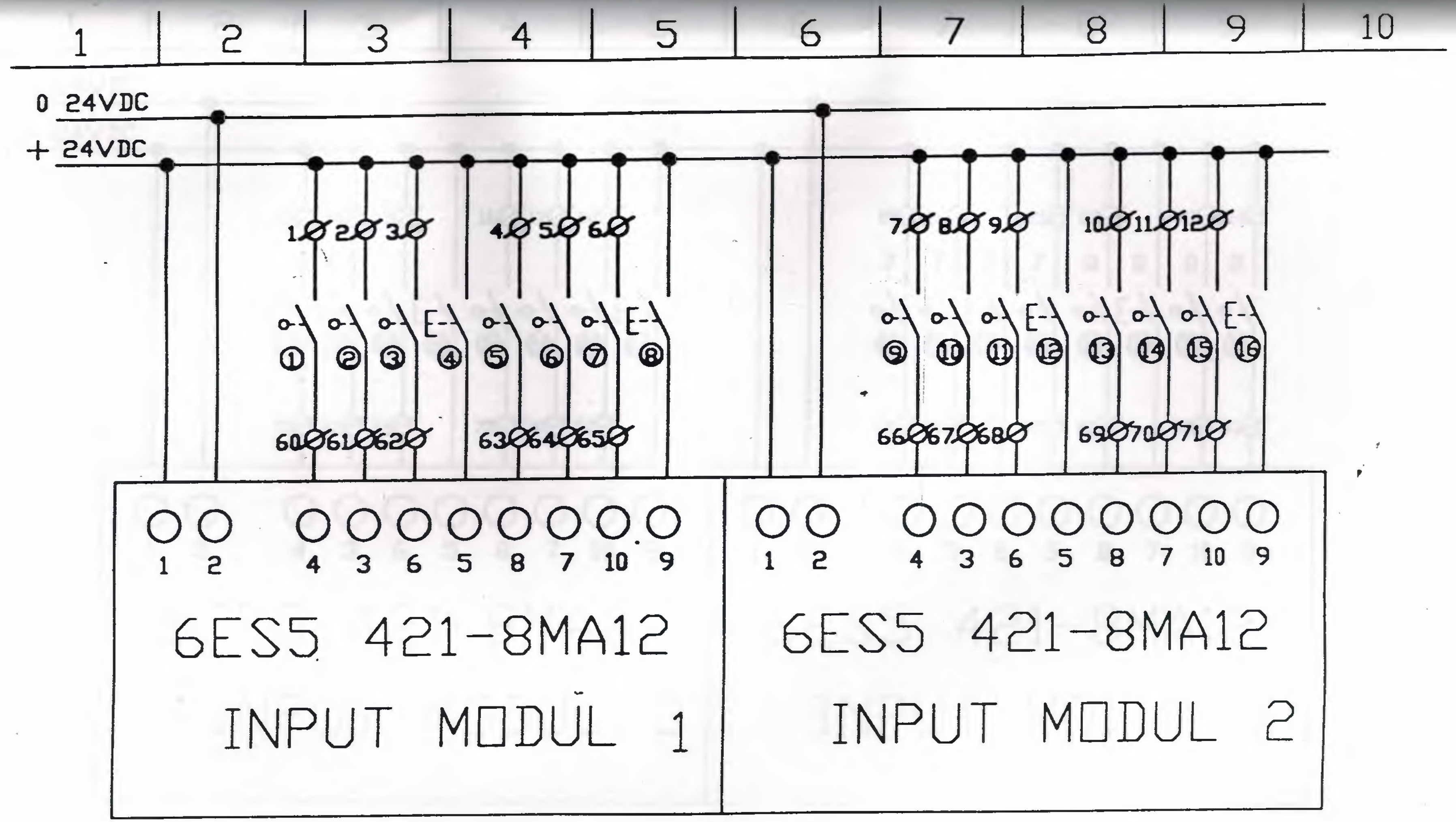
Yapan	Cizen	Kontrol	Tarih	IS NO: 53	Kod No
SUAT	SUAT	RIDVAN	03.02.95		95-53



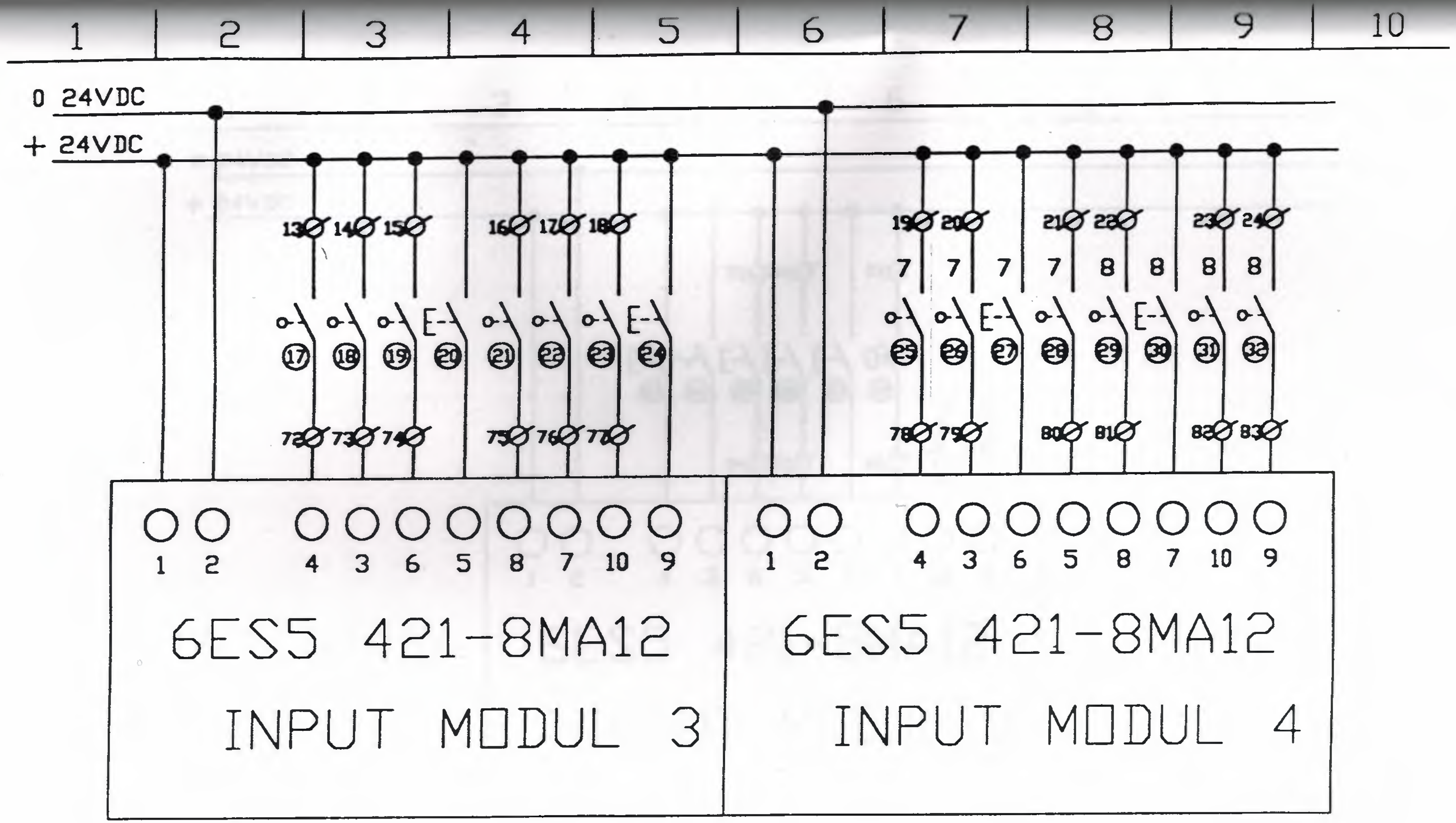
Yapan	Cizen	Kontrol	Tarih		IS NO: 53	Kod No
SUAT	SUAT	RIDVAN	03.02.95			95-53



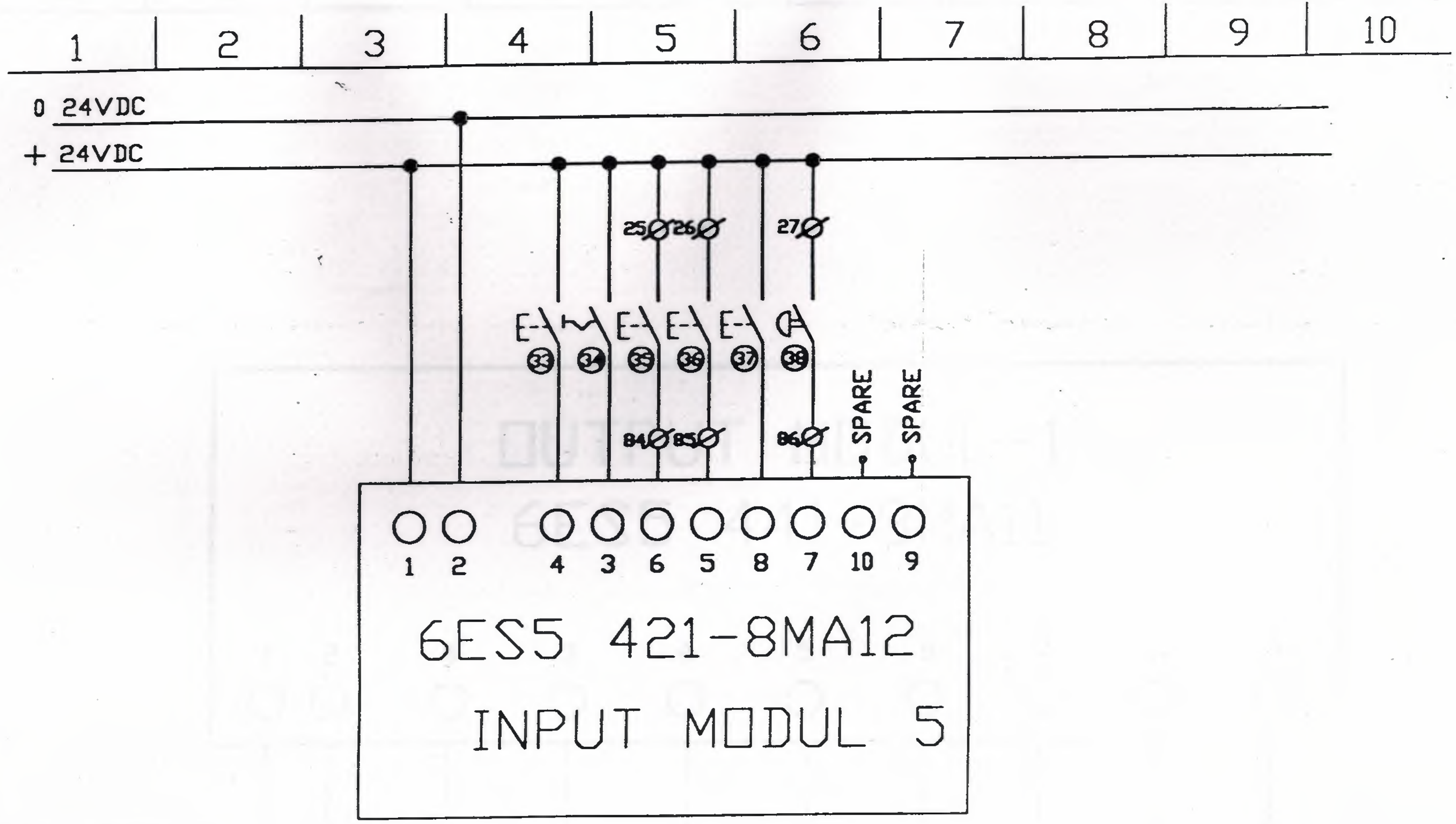
Yapan	Cizen	Kontrol	Tarih	IS NO: 53	Kod No
SUAT	SUAT	RIDVAN	03.02.95		95-53



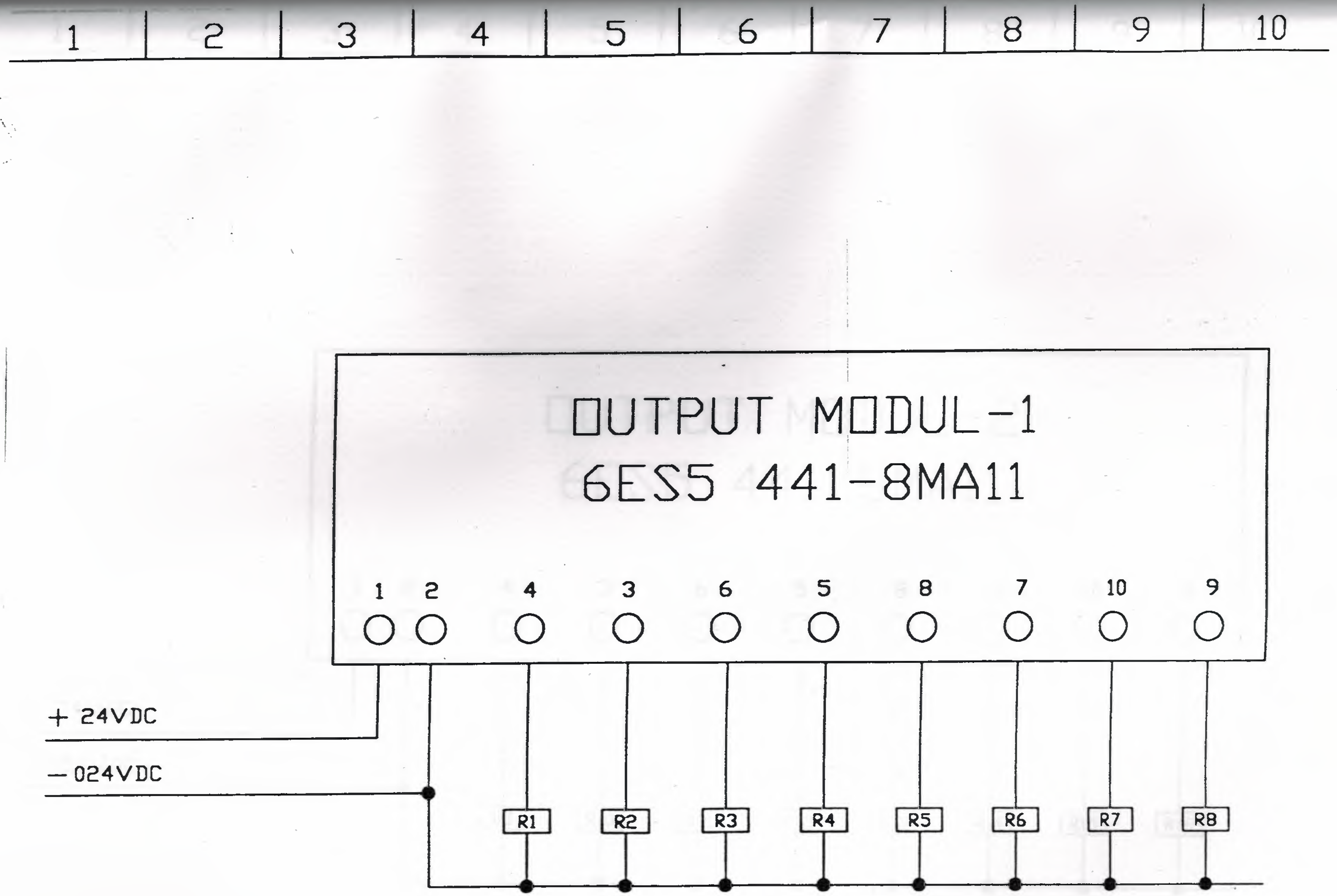
Yapan	Cizen	Kontrol	Tarih		IS NO: 53	Kod No
SUAT	SUAT	RIDVAN	06.02.95			95-53



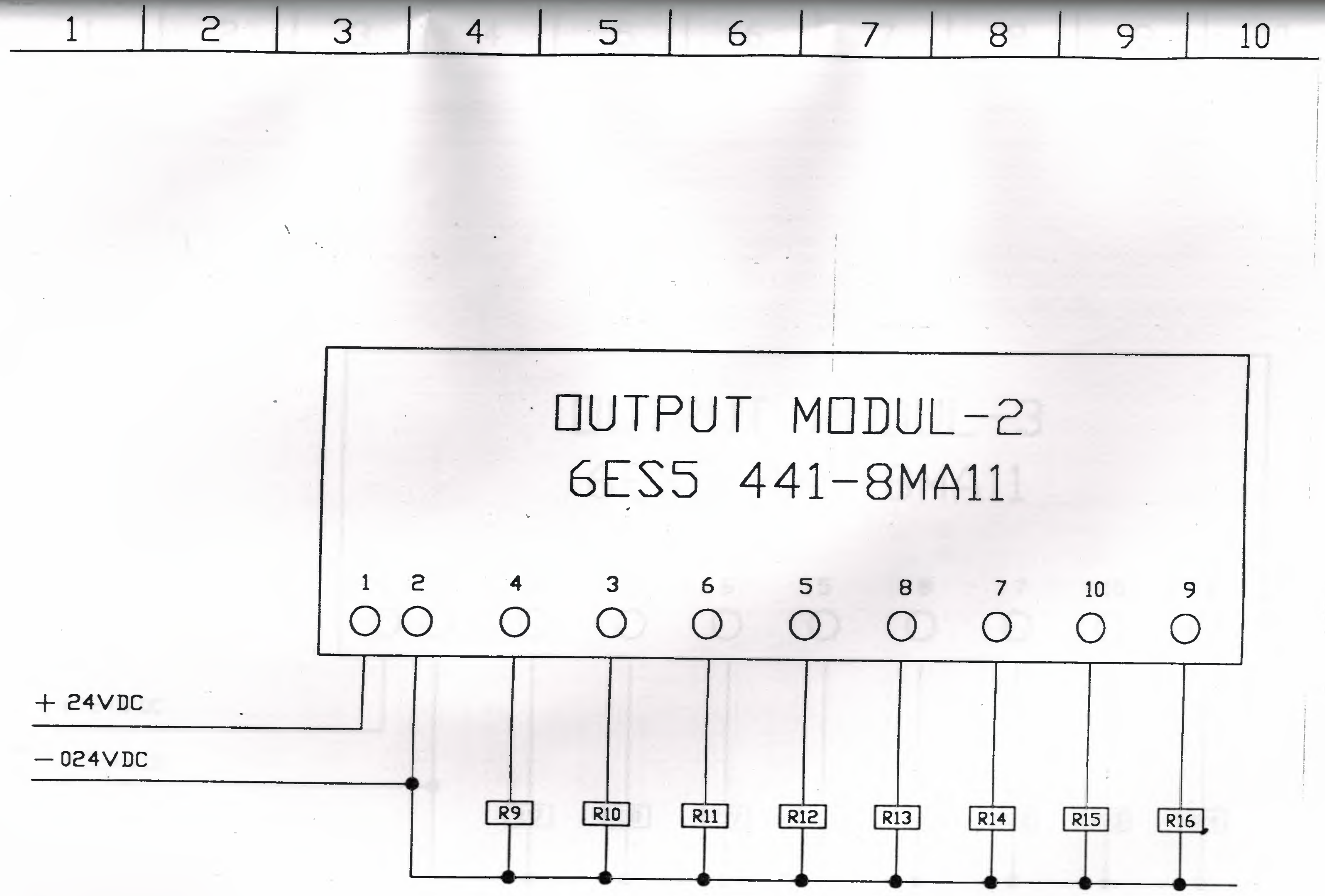
Yapan	Cizen	Kontrol	Tarih		IS NO: 53	Kod No
SUAT	SUAT	RIDVAN	06.02.95			95-53



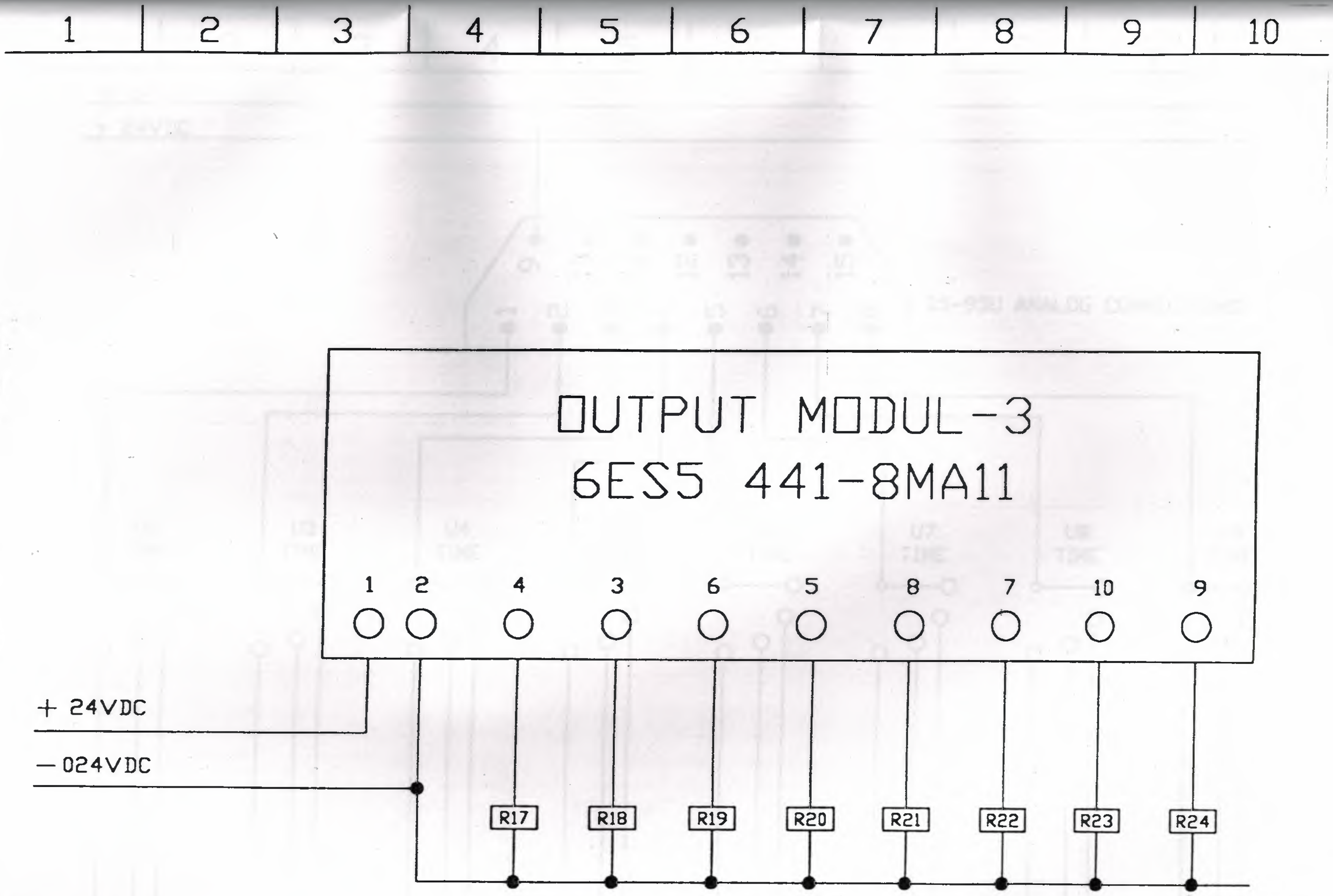
Yapan	Cizen	Kontrol	Tarih	IS NO: 53	Kod No
SUAT	SUAT	RIDVAN	06.02.95		95-53



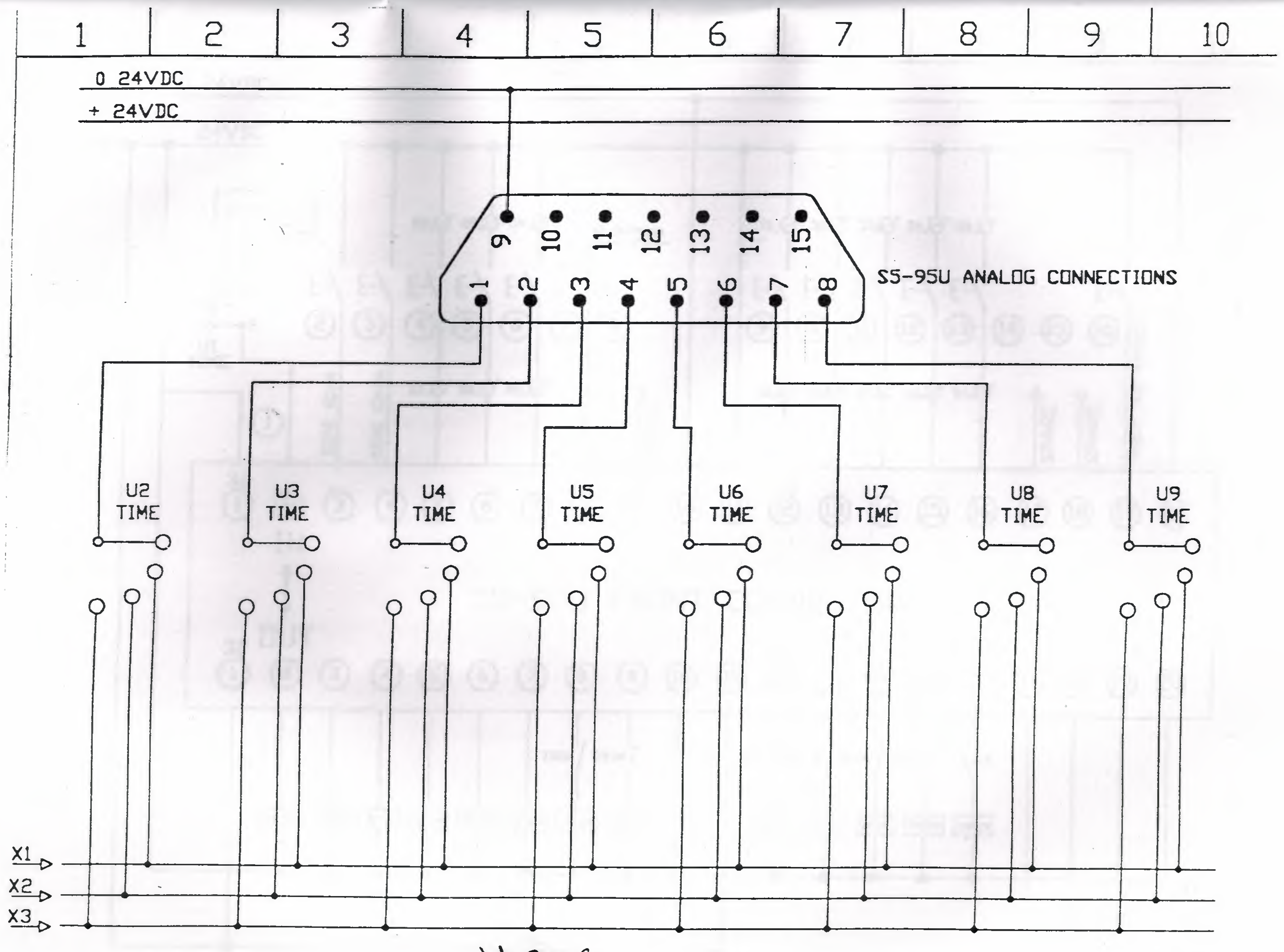
Yapan	Cizen	Kontrol	Tarih		IS NO: 53	Kod No
ERCU	ERCU	RIDVAN	24.02.95			



Yapan		Cizen		Kontrol		Tarih		IS NO: 53	Kod No
ERCU		ERCU		RIDVAN		24.02.95			



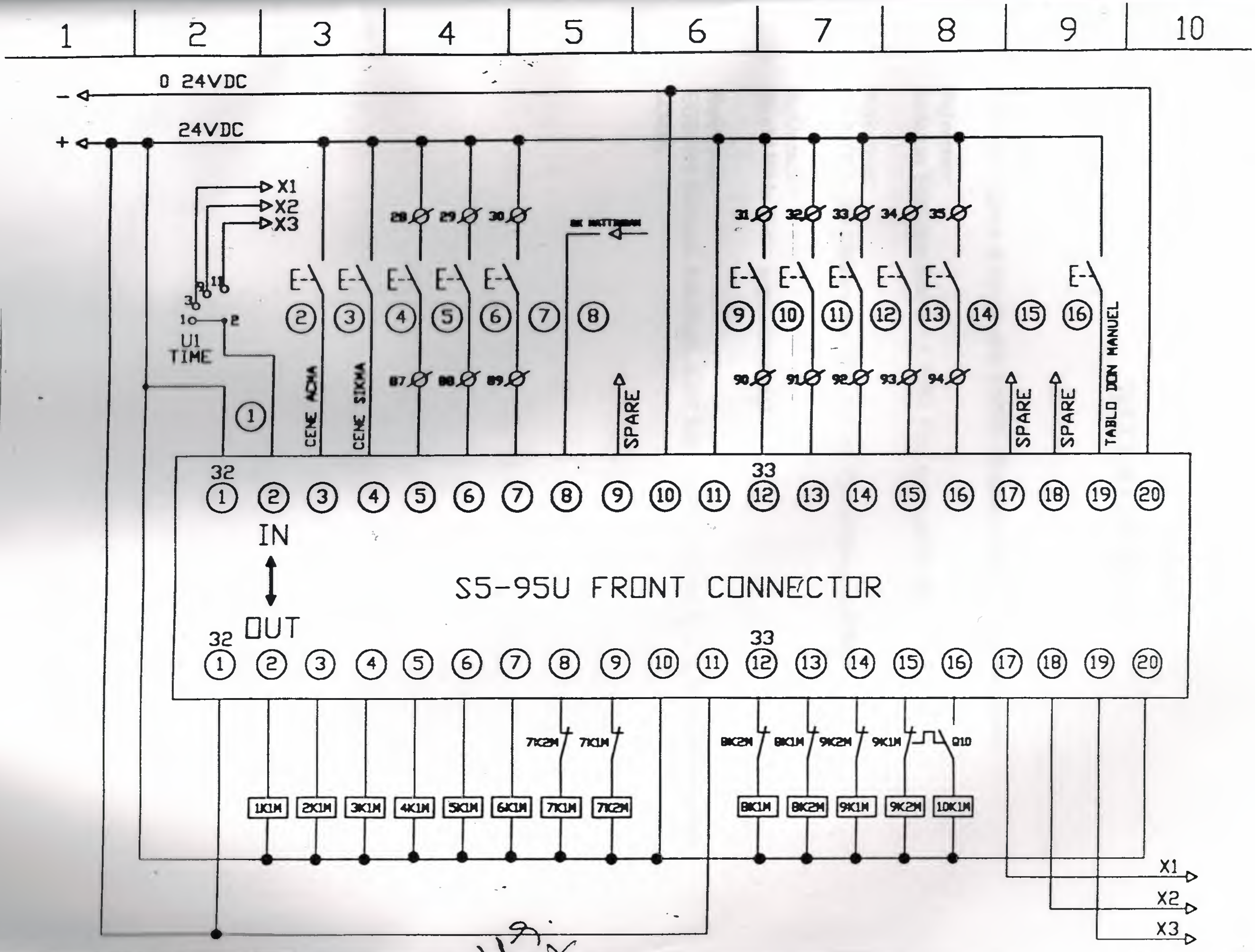
Yapan	Cizen	Kontrol	Tarih	IS NO: 53	Kod No
ERCU	ERCU	RIDVAN	24.02.95		



Yapan	Cizen	Kontrol	Tarih
RIDVAN	SUAT	RIDVAN	25.04.94

IS NO: 074

Kod No
95-53



Yapan	Cizen	Kontrol	Tarih	IS NO: 53	Kod No
RIDVAN	ERCU	RIDVAN	25.04.94		95-53

REFERENCES

Reference:
Mustafa Yağımlı & Feyzi Akar (1999). Programmable Logic Controllers.

Reference:
Erdoğan Teközgen İstanbul, (1998). PLC ve Uygulamaları

Reference:
HAKER Soğuk Döküm San. Tic. Lti. Şti. Tansel Sarıçam İZMİR

Reference:
World Wide Web: www.siemens.com

Reference:
EGESİM Siemens Ana Bayii. 1204 Sok. No:41/1-l Bulanalp 2 İş Merkezi Yenişehir İZMİR.