



# AUTOMATED CONTACTOR ASSEMBLY



A Project Report

p-2366

*Submitted by*

<b>R.Lavanya</b>	-	<b>71204105032</b>
<b>T.Saranya</b>	-	<b>71204105044</b>
<b>M.Surya</b>	-	<b>71204105056</b>



*in partial fulfillment for the award of the degree  
of*

**Bachelor of Engineering  
in  
Electrical & Electronics Engineering**

**DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING**

**KUMARAGURU COLLEGE OF TECHNOLOGY**

**COIMBATORE – 641 006**

**ANNA UNIVERSITY: CHENNAI 600 025**

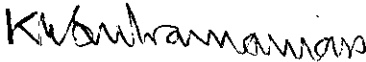
**APRIL– 2008**

## BONAFIDE CERTIFICATE

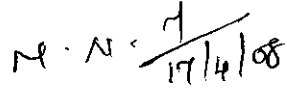
Certified that this project report entitled "AUTOMATED CONTACTOR ASSEMBLY" is the bonafide work of

Ms.R.Lavanya - Register No.71204105032  
Ms.T.Saranya - Register No.71204105044  
Ms.M.Surya - Register No.71204105056

who carried out the project work under my supervision.



Signature of the Head of the Department  
Prof.K.Regupathy Subramaniam



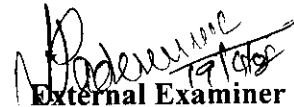
Signature of the Guide  
Mrs.M.Nirmala

71204105032, 71204105044, 71204105056

Certified that the candidate with university Register No. \_\_\_\_\_ was examined in project viva voce Examination held on 19.04.2008



Internal Examiner



External Examiner

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING  
KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE 641 006



LAKSHMI ELECTRICAL CONTROL SYSTEMS LIMITED

LECS/HR

27-03-2008

**TO WHOMSOEVER IT MAY CONCERN**

This is to certify that the following IV year BE (EEE) students of Kumaraguru College of Technology, Coimbatore have done a project work titled "**Automated Contactor Assembly**" in our organization from Sep 2007 till March 2008.

Name of the Students:

<b>Name:</b>	<b>Reg No:</b>
1) Ms. R.Lavanya	71204105032
2) Ms. T.Saranya	71204105044
3) Ms. M.Surya	71204105056

During their study, they were sincere and showed keen interest in data collection, and in interpretations.

We wish success in all their future endeavors.

For LAKSHMI ELECTRICAL CONTROL SYSTEMS LIMITED

  
(A.MURUGESAN)  
Manager - HRD

# ABSTRACT

## ABSTRACT

The main contacts (NO & NC) were inserted in the contactor manually in earlier days. By manually inserting the contacts in the contactor resulted in the production of about 20 pieces per hour. This leads to the disadvantage of more time consuming and less productivity. This project “**Automated contactor assembly**” automates all manual process.

Here’s the technical approach to reduce the manual interruption with the introduction of programmable logic controller [S7-226]. The advantages of automation using PLC are high-speed operation, more accuracy, and low break down, easy trouble shooting resulting in production of about 45 pieces per hour. Input is given through reed switch to PLC and output is taken using solenoid valve from PLC. As the main contact insertion is the only dedicated function automation using PLC results in more cost and also time consuming process.

To over come these disadvantages automation is done using PIC microcontroller [16F877A]. Now the production will be increased up to 60 pieces per hour. As the production increases, there will increase in the company’s profit

The contactor top housing component is obtained from the plastic molding department. The assembled main contacts, brass with welded silver is obtained from the metal shop. The contactor top housing is placed over the holding carriage and is fixed. The main contacts are placed in the contactor and fixed to the specified depth [3 to 4 mm] automatically.

# **ACKNOWLEDGEMENT**

## ACKNOWLEDGEMENT

The successful completion of our project can be attributed to the combined efforts made by us and the contribution made in one form or the other, by the individuals we hereby acknowledge.

We are highly privileged to thank **Dr. Joseph V. Thanikal**, B.E., M.E., Ph.D., PDF., CEPIT Principal, Kumarguru College of Technology for providing the necessary facilities for successful completion of our project.

We express our heartfelt gratitude and thanks to the Dean (Academics), **Dr. S. Sadhasivam** for encouraging us to do the project.

We wish to express our deep sense of gratitude and indebtedness to the Dean (R&D) and Head of Department of Electrical and Electronics Engineering, **Dr. K. Regupathy Subramaniam**, B.E.(Hons), M.Sc., for his enthusiasm and encouragement, which has been instrumental in the success of the project.

We wish to place on record our deep sense of gratitude and profound thanks to our guide, **Mrs. L. Nirmala** M.E., lecturer, Department of Electrical and Electronics Engineering, for her valuable guidance, constant encouragement, continuous support and co-operation rendered throughout the project.

We thank **Mr. Nagarajan** and **Mr. Elango** who gave this opportunity to do this project in Lakshmi Electrical and Control systems (LECS).

We are also thankful to all teaching and non teaching staffs of Department of Electrical and Electronics Engineering for their kind help and encouragement in making our project successful.

We extend our sincere thanks to all our parents and friends who have contributed their ideas and encouraged us for completing the project.

# CONTENTS

<b>Title</b>	<b>Page No.</b>	
Bonafide Certificate	ii	
Certificate of the Company	iii	
Abstract	v	
Acknowledgement	vii	
Contents	viii	
List of Tables	xi	
List of Figures & Graphs	xii	
List of Abbreviations	xiii	
 <b>CHAPTER 1. INTRODUCTION</b>		
1.1	Need for the project	2
1.2	Objective	2
1.3	Contacts insertion using PLC	2
1.4	Contacts insertion using micro controllers	3
 <b>CHAPTER 2. CONTACTS INSERTION PROCESS</b>		
2.1	Sequence of the process	5
 <b>CHAPTER 3. PROGRAMMABLE LOGIC CONTROLLER</b>		
3.1	Introduction	11
3.1.1	Advantages of PLC	11
3.1.3	S7-200 Micro PLCs	13
3.1.4	S7-200 Models	13
3.1.5	Basic PLC operations	14
3.2	Features	15
3.2.1	PLC features	15
3.2.2	S7-226 Model Features	15
3.3	Programming using ladder logic	16
3.3.1	Rules for Construction of a Ladder Diagram	16
3.3.2	Ladder logic diagram	16



3.2.2	S7-226 Model Features	15
3.3	Programming using ladder logic	16
3.3.1	Rules for Construction of a Ladder Diagram	16
3.3.2	Ladder logic diagram	16
3.3.3	Subroutines	16
3.3.4	Testing a Program	16
3.4	PLC Program	22
3.5	PLC tools	29
3.5.1	STEP 7-MICRO/WIN Software	29
	3.5.1.1 Downloading and installation instruction	29
3.5.2	Features of software	30
	3.5.2.1 Time Saving data Entry	30
	3.5.2.2 Password Protection	30
	3.5.2.3 Uniform Basic Functionality	30
	3.5.2.4 Project Documentation	30

## **CHAPTER 4. AUTOMATION USING EMBEDDED SYSTEM**

4.1	Introduction	32
4.2	Structure of embedded system	33
4.3	Features	35
4.3.1	Micro controller Core Features	35
4.3.2	Peripheral Features	35
4.3.3	Embedded system features	37
4.4	Microcontroller	37
4.4.1	Parallel ports	37
4.4.2	Pull up resistor	38
4.4.3	Timer/counter	38
4.4.4	Downloading	38
4.4.5	Debugging technique	39

4.6.1.1 Features of compiler	54
4.6.1.2 Additional features	55

## **CHAPTER 5. COMPARISON RESULT**

## **CHAPTER 6. CONCLUSIONS AND RECOMMENDATIONS**

6.1	Conclusions from the project	60
6.2	Recommendations for future work	61

<b>BIBLIOGRAPHY</b>	<b>63</b>
---------------------	-----------

## LIST OF TABLES

<b>Table</b>	<b>Title</b>	<b>Page No.</b>
1.1	S7-226 Features	13
1.2	Input for PLC	17
1.3	Output for PLC	17
1.4	Embedded input	42
1.5	Embedded output	42

## LIST OF FIGURES

<b>Figure</b>	<b>Title</b>	<b>Page No.</b>
1.1	Flowchart for contacts insertion process	6
1.2	PLC block diagram	12
1.3	PLC module	14
1.4	PLC schematic diagram	18
1.5	Flowchart for PLC	19
1.6	Architecture of PIC 16F877A	40
1.7	Pin diagram of PIC 16F877A	41
1.8	Embedded schematic diagram	43
1.9	Flowchart for embedded	44
1.10	Comparison results	58

## LIST OF ABBREVIATIONS

ALU	-	Arithmetic Logic Unit
ADC	-	Analog to Digital Converter
ACK	-	Acknowledgement
BOR	-	Brown Out Reset
CMOS	-	Complementary Metal Oxide Semiconductor
FBD	-	Function block diagram
EEPROM	-	Electrically Erasable Programmable Read Only Memory
GPR	-	General Purpose Register
MCLR	-	Master Clear
PWM	-	Pulse Width Modulation
LAD	-	Ladder logic diagram
STL	-	Statement list
POP	-	Power On Reset
RAM	-	Random Access Memory

**Chapter-1**

**INTRODUCTION**

## **1. INTRODUCTION**

Automation has been assumed to be an increasingly important role in the development and advancement of modern civilization and technology. Contactor is used to start the motor. The contactor consists of normally opened and normally closed (NO & NC) contacts, which are needed to be inserted in the contactor and before this preinsertion, is done to a depth of about 4 mm. The assembled main contacts are made up of brass welded with silver and the contactor is made up of plastic.

### **Existing method**

In the existing system the contacts were pre-inserted into the contactor by manual operation. The operator pushes the cylinders manually used for this purpose backward and forward. It gives only very less production.

### **1.1 OBJECTIVE**

To insert the main contacts in the contactor automatically using PLC & PIC microcontroller to increase the productivity.

### **1.2 NEED FOR THE PROJECT**

This project is need to over come the drawbacks in the existing method. The disadvantages of existing methods are,

1. Less production.
2. More time consuming.
3. Manually the insertion to exact depth is also not possible.
4. The contacts may get inserted in the reverse position.
5. As the contactor is made up of plastic, if excess pressure is applied during the process may result in the breakage of the contactor.

### **1.3 CONTACTS INSERTION USING PLC**

The PLC is required as the automation done using it results in more production that is about forty-five pieces is produced per hour. This is a dedicated function. The PLC has become a standard for control. It now not only replaces the earlier relay controls but

has also taken over many additional control functions. It is possible to implement control tasks in the form of software programs.

The S7-200 Micro PLC is a smallest member of the SIMATIC S7 family of programmable controllers. The Central Processing unit (CPU) is internal to the PLC. The inputs and outputs are the system control points. Input monitor field devices such as switches and sensors. Outputs control other devices such as motors and pumps. The programming ports are the connection to the programming device.

There are four S7-200 CPU types:

- S7-221
- S7-222
- S7-224
- S7-226

For this project S7-226 model is used.

## **1.4 CONTACTS INSERTION USING MICROCONTROLLERS**

A single chip Microcontroller is obtained by integrating all the components of a Microcomputer in one IC package. Hence apart from CPU such a single-chip microcontroller will therefore contain its own clock generators and some amount of ROM or EPROM, RAM and I/O channels etc on the same chip.

### **1.4.1 PIC Micro Controllers**

The microcontroller that has been used for this project is from PIC series. PIC microcontroller is the first RISC based microcontroller fabricated in CMOS (complimentary metal oxide semiconductor) that uses separate bus for instruction and data allowing simultaneous access of program and data memory.

### **1.4.2 PIC (16F877A)**

Various microcontrollers offer different kinds of memories. EEPROM, EPROM, FLASH etc of which FLASH is the most recently developed. Technology that is used in pic16F877A is flash technology, so that data is retained even when the power is switched off. Easy Programming and Erasing are other features of PIC 16F877A.



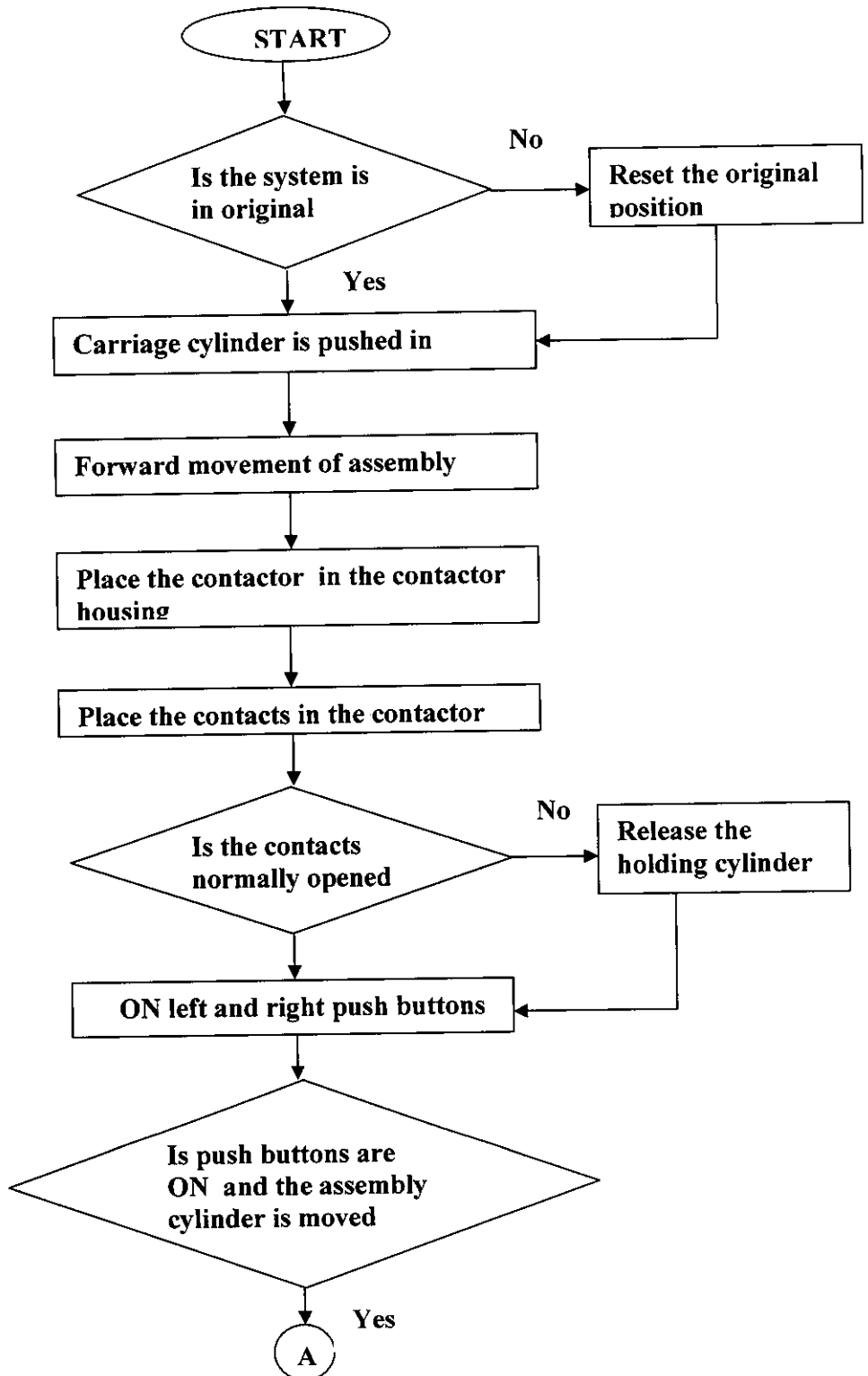
**Chapter-2**  
**PROCESS**

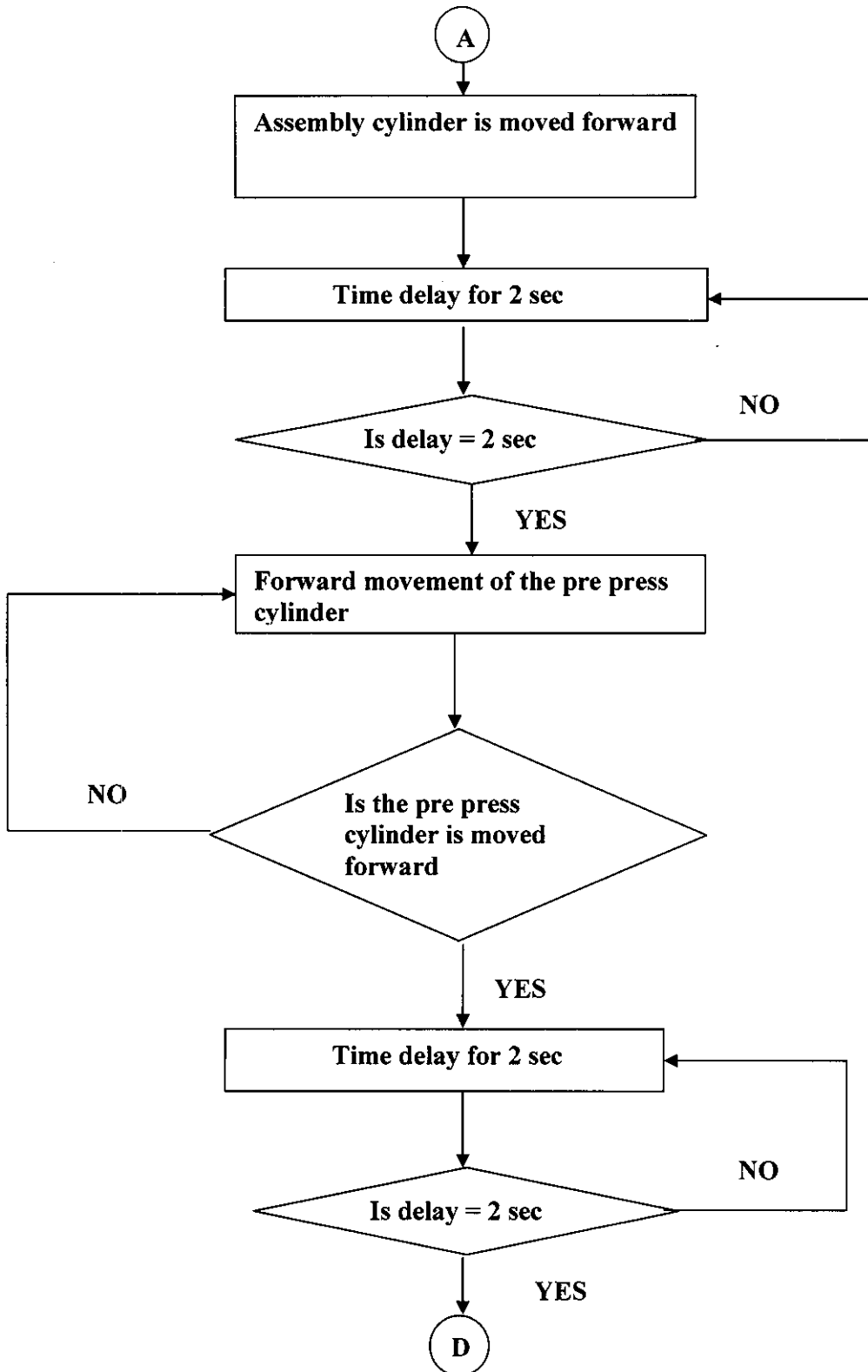
## **2. CONTACTS INSERTION PROCESS**

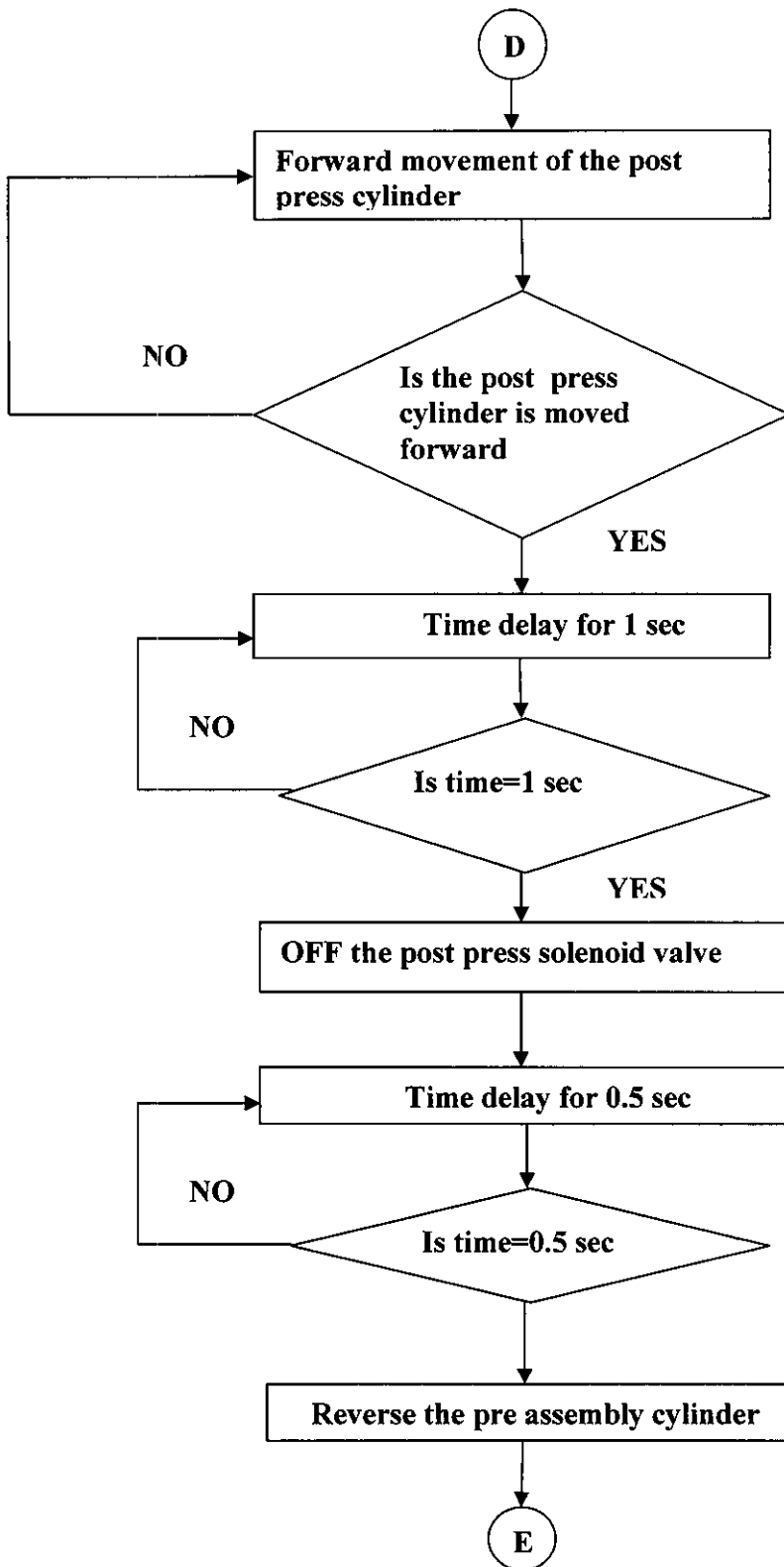
### **2.1 Sequence of the process**

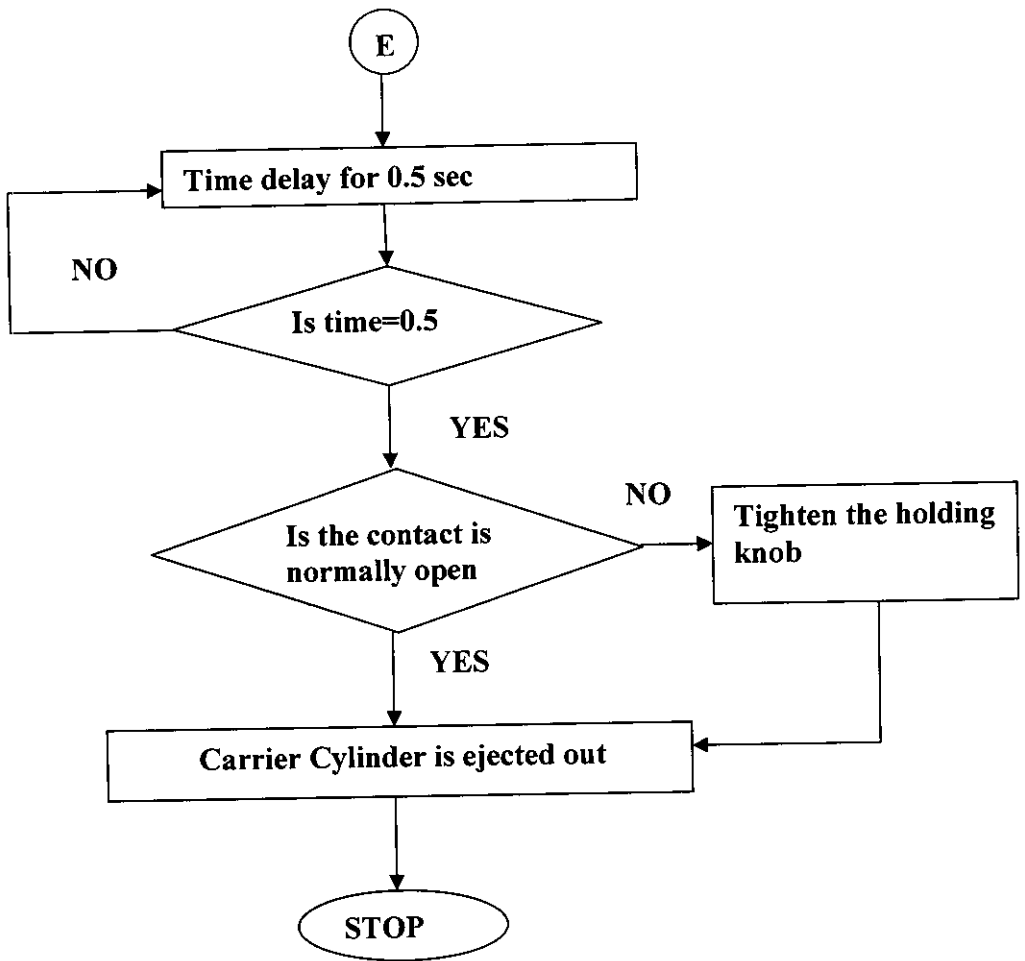
- ❖ Forward movement of the carriage cylinder.
- ❖ Assembly cylinder moves forward and holds the contactor housing placed by the operator.
- ❖ Release the holding knob [Not necessary for normally opened contacts].
- ❖ In the time interval of about 0.5sec pre-press cylinder moves forward and pre fixes the contacts.
- ❖ Followed by the pre-press cylinder, post-press cylinder moves and fixes the contacts to the required depth.
- ❖ Now the post-press cylinder moves in the reverse direction.
- ❖ Pre-press cylinder follows the post-press cylinder.
- ❖ Tighten the holding knob. [Not necessary for normally opened contacts].
- ❖ Assembly cylinder moves backward.
- ❖ Finally carriage cylinder is ejected and the contactor is taken out.

### 1.1 Flowchart for contacts insertion process









**Chapter-3**  
**PROGRAMMABLE**  
**LOGIC**  
**CONTROLLER**

## **3. PROGRAMMABLE LOGIC CONTROLLERS**

### **3.1 INTRODUCTION**

Control engineering has evolved over time. In earlier days, a system is controlled by manual operations. The advent of the PLC began in the 1970s, and has become the most common choice for manufacturing controls. PLCs have been gaining popularity on the factory floor and will probably remain predominant for some time to come. There is a constant need for process control systems in the manufacturing industry to produce the better quality product more efficiently and at a low cost. This has led to the evolution of automated system.

#### **3.1.1 Advantages of PLC**

- Cost effective for controlling complex systems.
- Flexible and can be reapplied to control other systems quickly and easily.
- Computational abilities allow more sophisticated control.
- Trouble shooting aids make programming easier and reduce downtime.
- Reliable components make these likely to operate for years before failure.

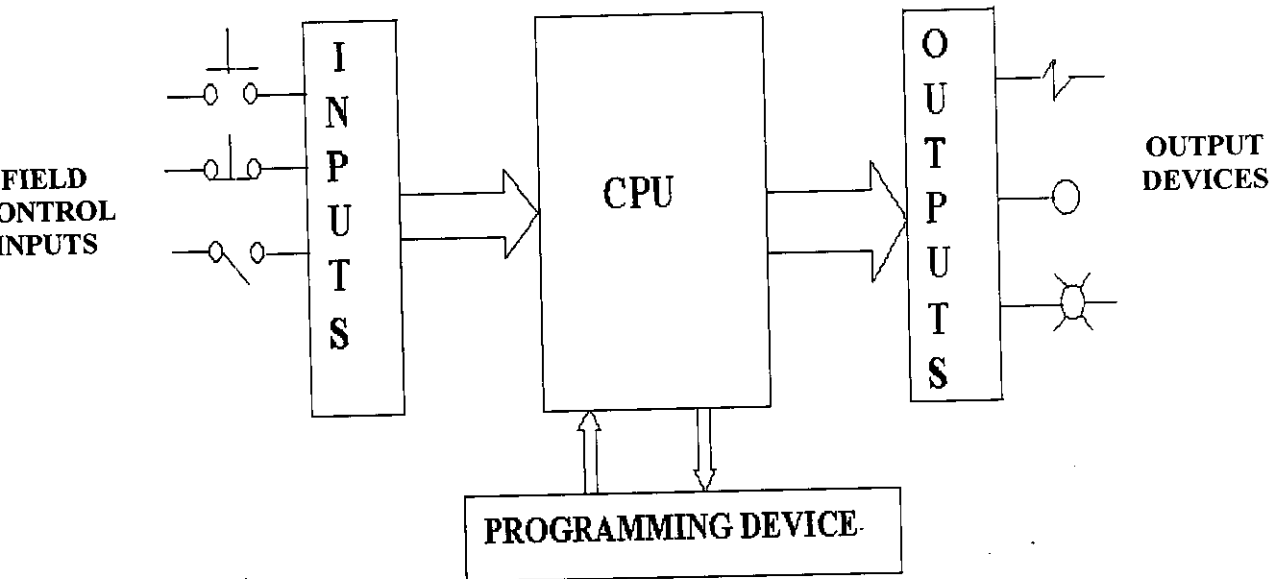
#### **3.1.2 PLC block diagram**

PLC consists of the following essentials

- Input modules
- CPU with processor and program memory
- Output modules
- Bus system
- Power supply







**Fig 3.1: PLC block diagram**

### **Input module**

Acts as an interface between the field control inputs and the CPU. The voltage or current signals generated by the sensors, transducers, limit switches, push buttons etc are applied to the terminals of the input module. It converts the field signal into a standard control signal for processing by the CPU.

### **Output module**

Acts as a link between the CPU and the output devices located in the field. The field devices could be relays, contactors, lamps, actuators solenoid valves, etc., the output module converts the output signal delivered by the CPU into an appropriate voltage level suitable for the output field device.

## CPU

It consists of the following

- ❖ Arithmetic logic unit
- ❖ Program memory
- ❖ Process image memory
- ❖ Internal registers and counters
- ❖ Flags
- ❖ Heart of the CPU is its microprocessor or microcontroller.

### 3.1.3 S7-200 Micro PLCs

The S7-200 Micro PLC is the smallest member of the SIMATIC S7 family of Programmable controllers. The Central Processing Unit (CPU) is internal to the PLC. Input unit and output unit (I/O) are the system control points. Input unit monitor field devices such as switches and sensors. Output unit control other devices such as motors and pumps. The programming port is the connection to the programming device.

### 3.1.4 S7-200 Models

There are four S7-200 CPU types: S7-221, S7-222, S7-224 and S7-2226 and three-supply configuration for each type. For this project “Automated contactor assembly” S7-226 model is used as it has the required number of inputs and outputs.

<i>Model description</i>	<i>Power supply</i>	<i>Input types</i>	<i>Output types</i>
226 DC/DC/DC	20.4-28.8 V <sub>DC</sub>	24 DC Inputs	16 DC Outputs
226 AC/DC/Relay	85-264 V <sub>AC</sub> 47-63 Hz	24 DC Inputs	15 Relay Outputs

**Table:3.1 S7-226 features**

### 3.1.5 Basic PLC operations

PLCs consist of input modules or points, a Central Processing unit (CPU), and output modules or points. An input accepts a variety of digital or analog signals from various field devices (sensors) and converts them into a logic signal that can be used by the CPU. The CPU makes decisions and executes control instructions based on program instructions in memory. Output modules convert control instructions from the CPU into a digital or analog signal that can be used to control various field devices (actuators). A programming device is used to input the desired instructions. These instructions determine the action of PLC for a specific input. An operator interface device allows process information to be displayed and new control parameters to be entered.

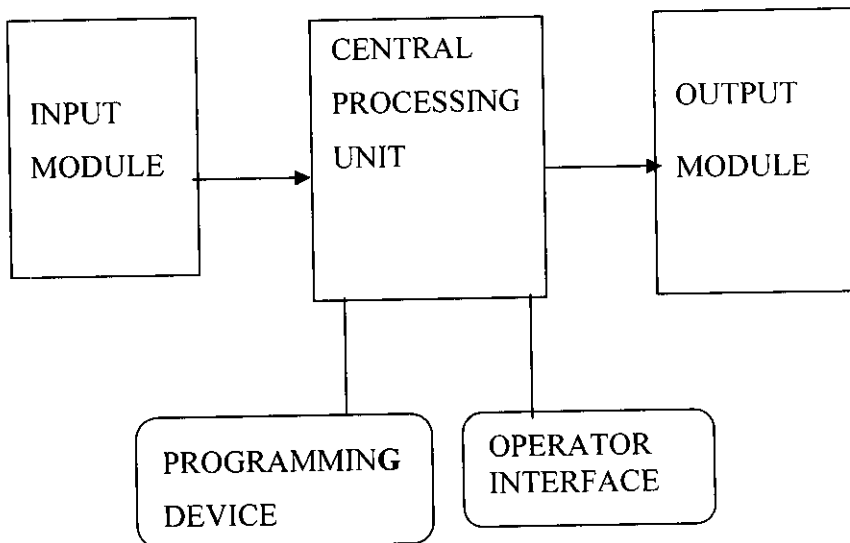


Fig 1.3: PLC Module

## 3.2 FEATURES

### 3.2.1 PLC Features

- ❖ Logic control
- ❖ PID control
- ❖ Operator control
- ❖ Coordination and communication
- ❖ Signaling and listing etc.,

### 3.2.2 S7-226 Model Features

Integrated Inputs/Outputs	24 DI/16 DO
Integrated Analogue In/Out	-
Max. Expansion modules	7
Max. # Of Dig. I/O channels	248
Analogue Points In/Out/Max	28/14/35
Program	16/24 KB
Data Memory	10 KB
Boolean Execution Time	0.22 Us
Bit Relays/Counters/Timers	256/256/256
High Speed Counters	6*30 KHz
Real Time Clock	Integrated
Pulse Output	2*20 KHz
Communication Interface	2*RS-485
Analogue Potentiometers	2

## **3.3 PROGRAMMING USING LADDER LOGIC**

### **3.3.1 Rules for Construction of a Ladder Diagram**

- ❖ All contacts must run horizontally, no vertically oriented contacts are allowed.
- ❖ The number of contacts per matrix is limited.
- ❖ Only one output may be connected to a group of contacts.
- ❖ Flow from left to right.
- ❖ Contact progression should be straight across.

### **3.3.2 Ladder logic diagram**

The left vertical line of a ladder logic diagram represents the power or energized conductor. The output element or instruction represents the neutral or return path of the circuit. The right vertical line, which represents the return path on a hard-wired control line diagram, is omitted. Ladder logic diagrams are read from left to right, top to bottom. Rungs are sometimes referred to as networks. A network may have several control elements, but only one output coil.

### **3.3.3 Subroutines**

Subroutines help to partition the program. The instructions used in the main program determine the execution of the specific subroutine. When the main program calls the subroutine for execution, the subroutine carries out its program to its end. Then, the system returns control to the main program at the network from which the subroutine was called.

### **3.3.4 Testing a Program**

Program is to be tested and debugged. The program is first downloaded from the programming device to the CPU. The selector switch is placed in the RUN position. The simulator switches are operated and the resulting indication is observed on the output indicator lamps.

## INPUT\_OUTPUT SWITCHES

<b>Input</b>	<b>Parts</b>	<b>Switches</b>
I <sub>0.0</sub>	Carriage Forward Cylinder	LS1
I <sub>0.1</sub>	Left Push Button	LS2
I <sub>0.2</sub>	Right Push Button	LS3

Table 1.2: Input for PLC

<b>Output</b>	<b>Parts</b>	<b>Switches</b>
Q <sub>0.0</sub>	Carriage Ejection Cylinder	SV1
Q <sub>0.1</sub>	Post Press Cylinder	SV2
Q <sub>0.2</sub>	Left Pre Press Cylinder	SV3
Q <sub>0.3</sub>	Right Pre Press Cylinder	SV4
Q <sub>0.4</sub>	Left Assembly Cylinder	SV5
Q <sub>0.5</sub>	Right Assembly Cylinder	SV6

Table 1.3: Output for PLC

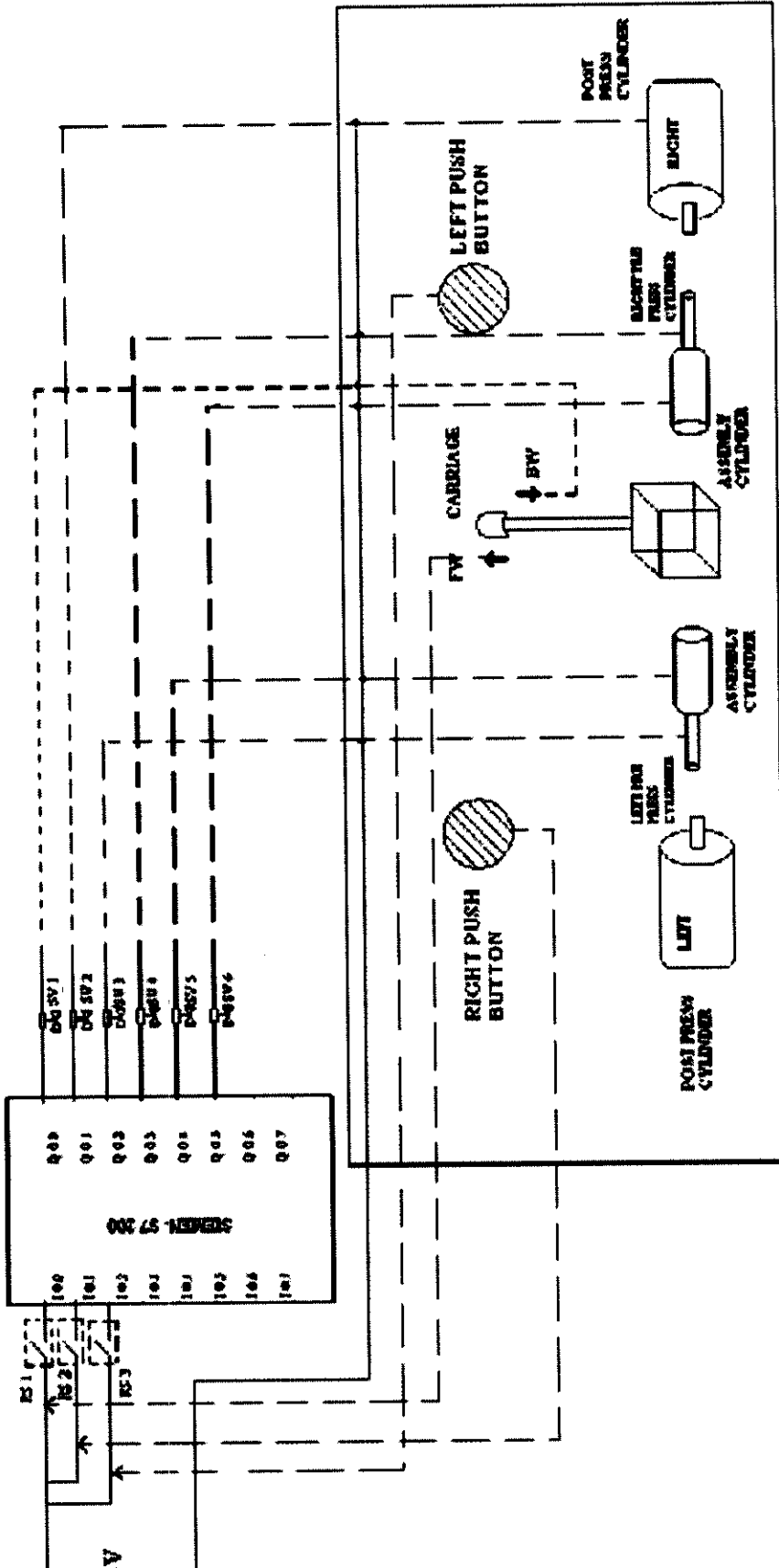
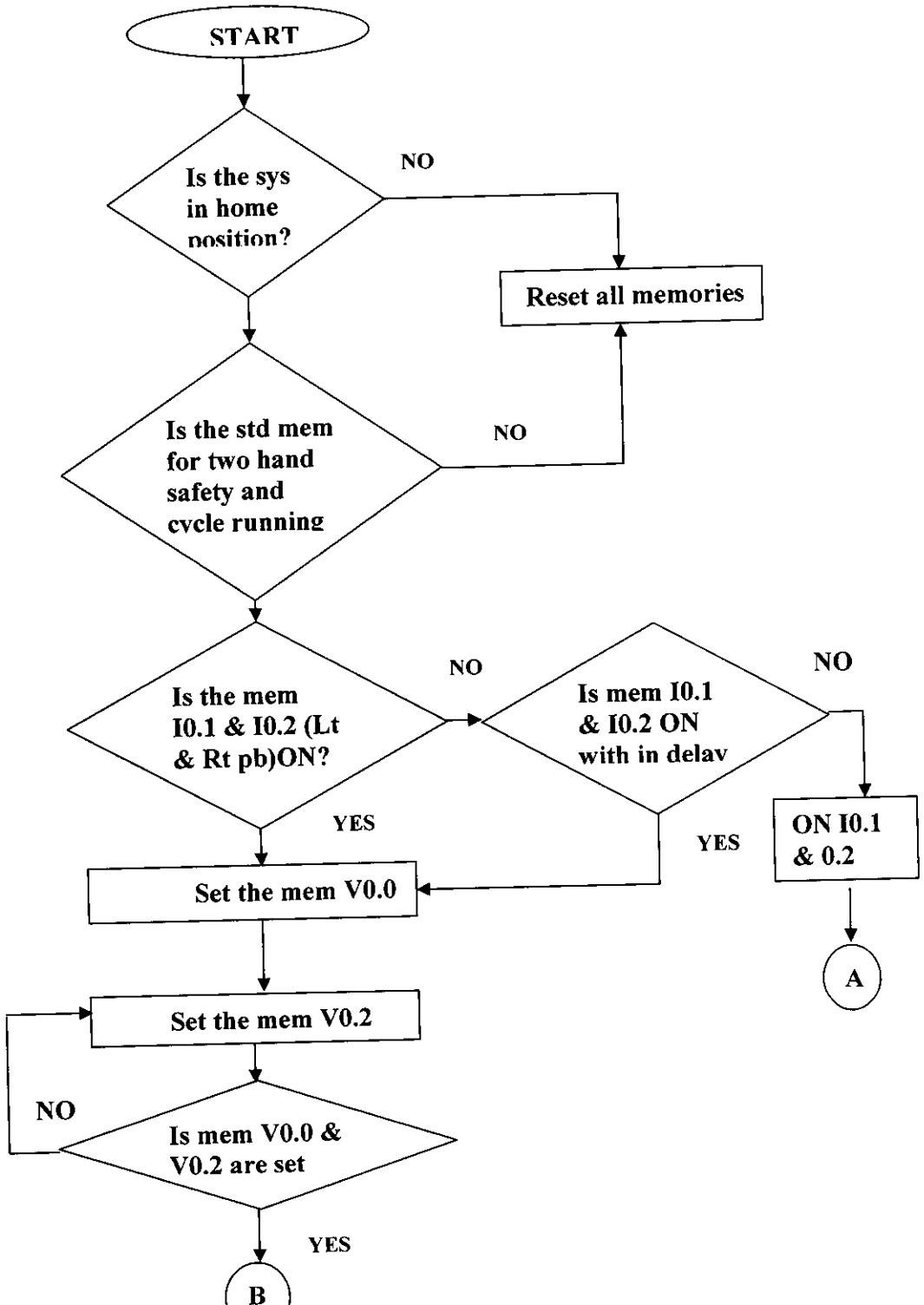
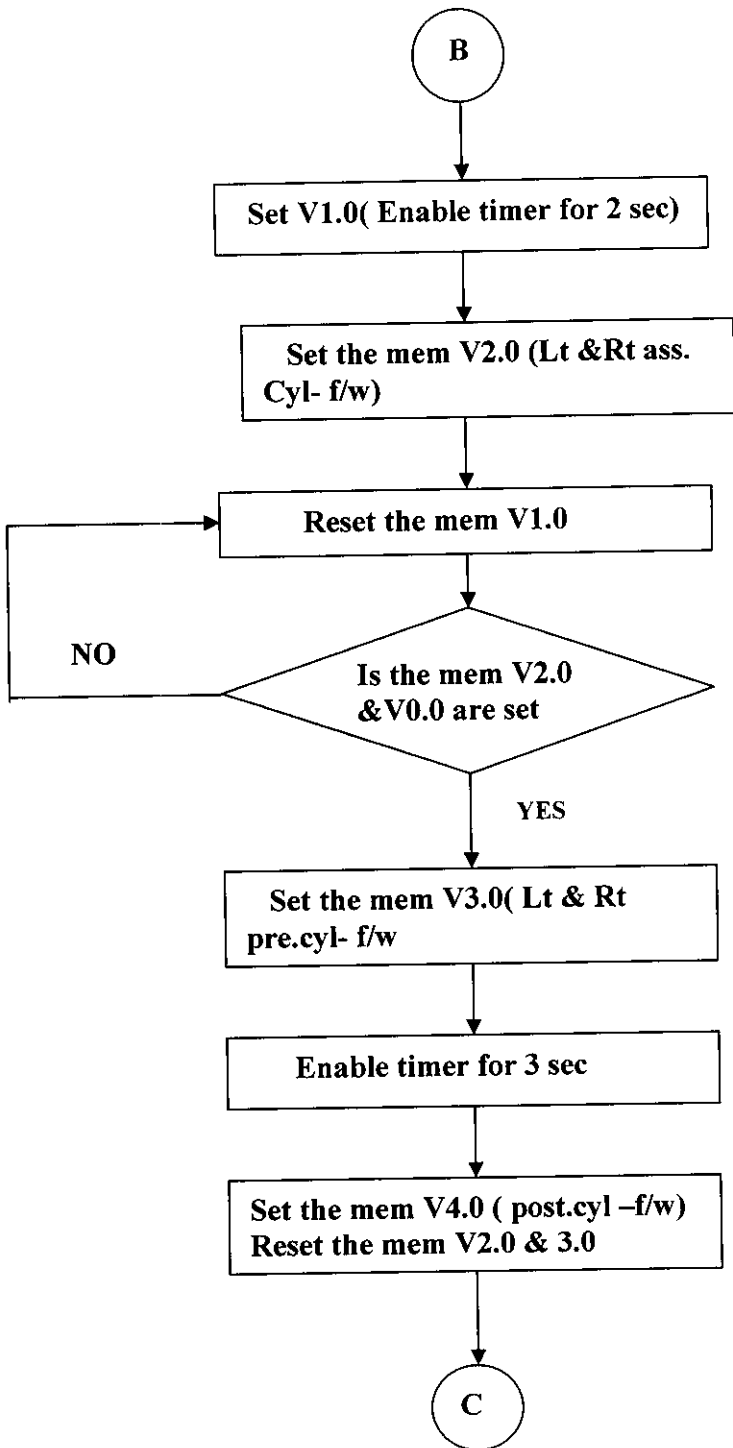


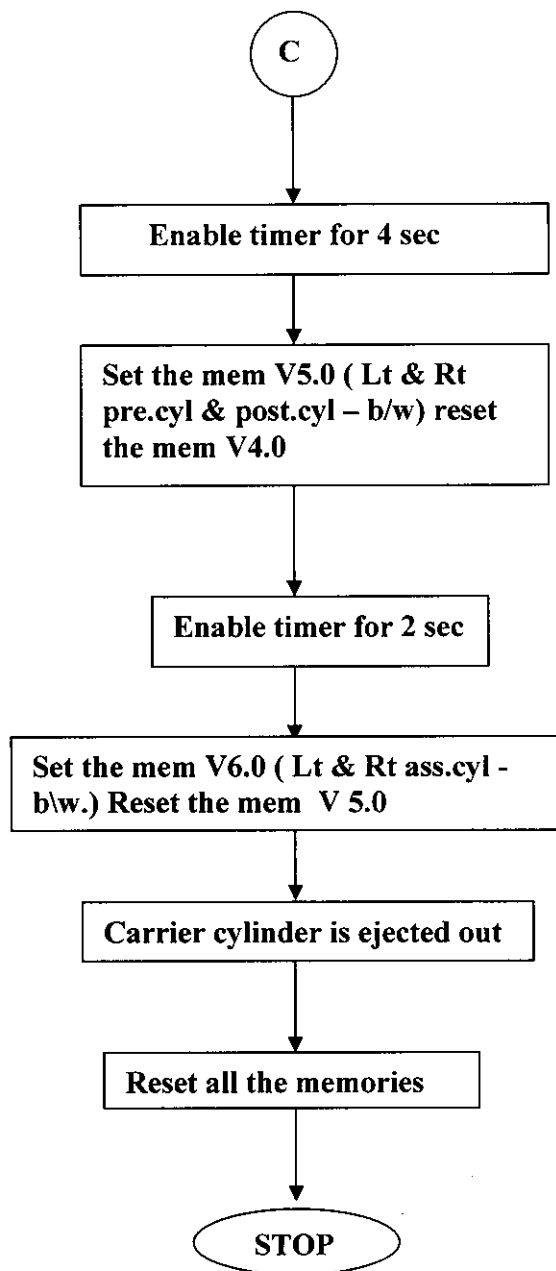
Fig. 3.3 Automated contactor assembling using PLC (S7-226)

Fig 1.5.Flow chart for PLC process:



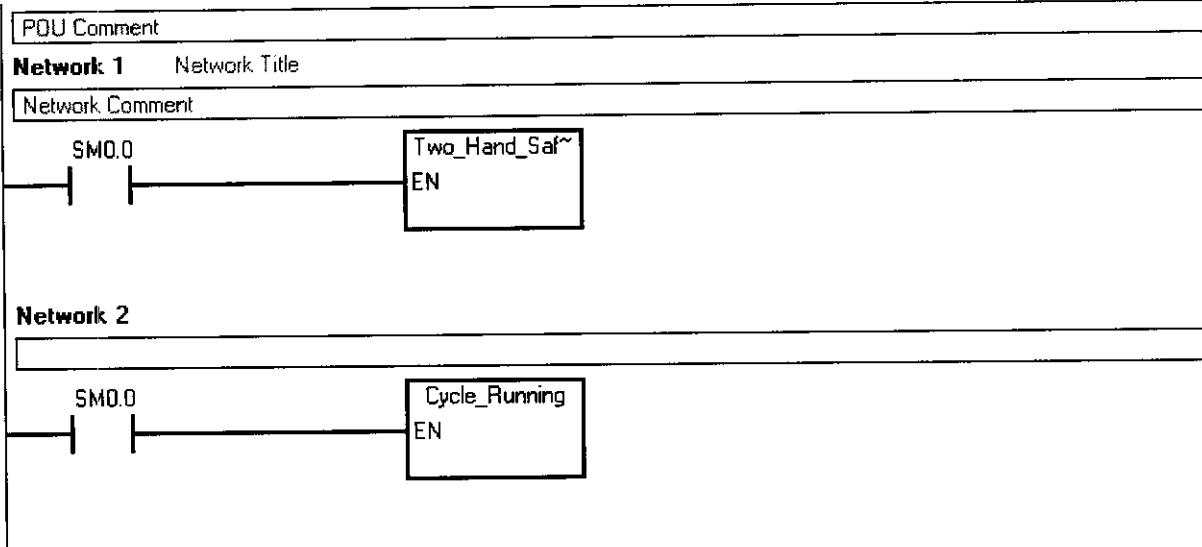




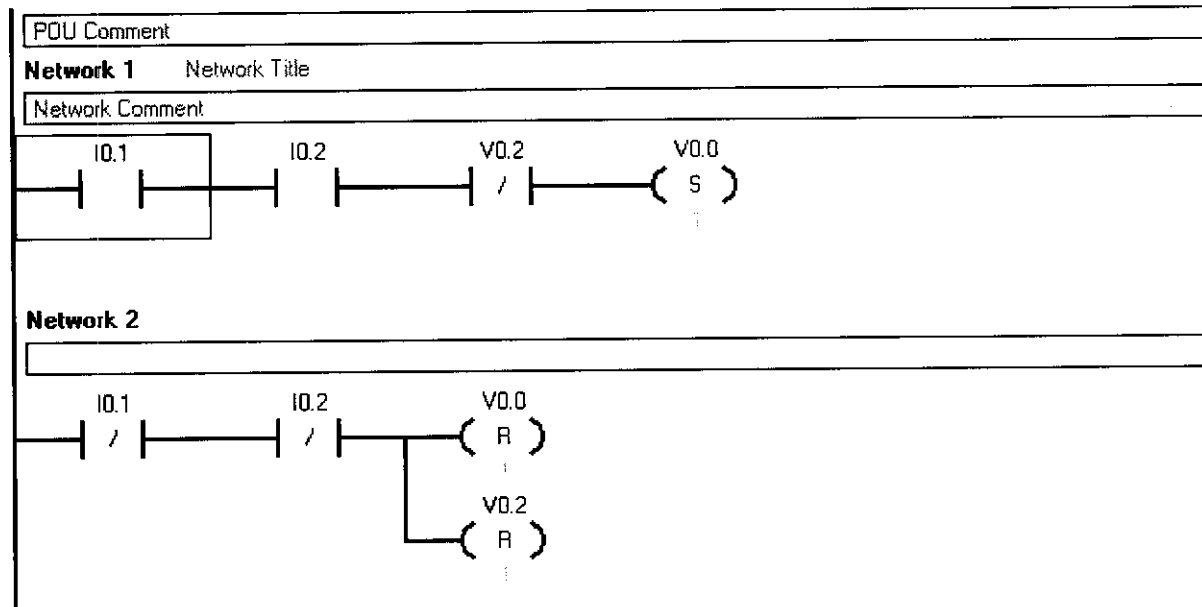


### 3.4 PLC PROGRAM

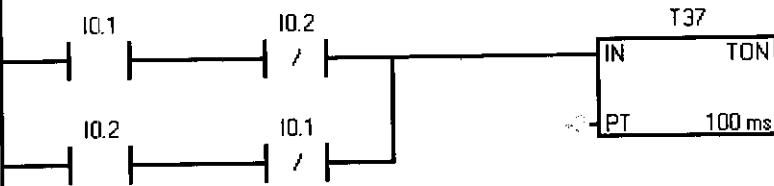
#### MAIN



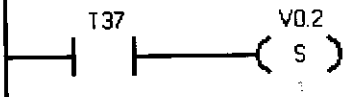
#### TWO HAND SAFETY



### Network 3

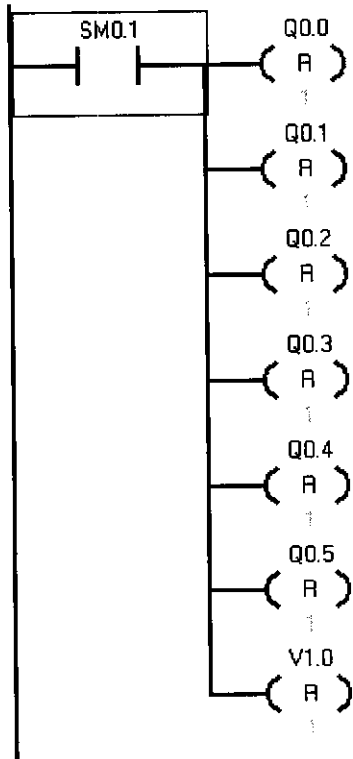


### Network 4

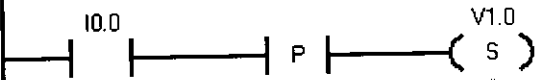


## CYCLE RUNNING

### Network 1



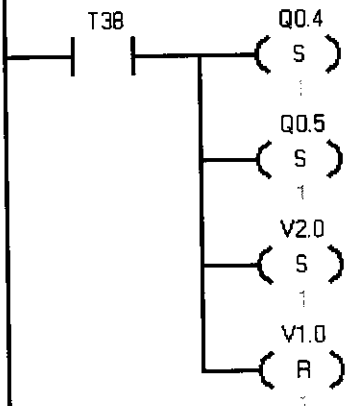
**Network 2**



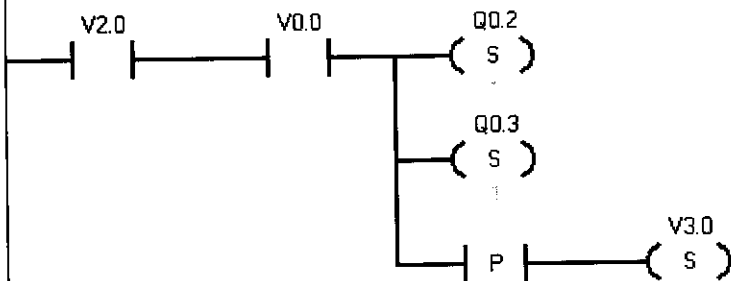
**Network 3**



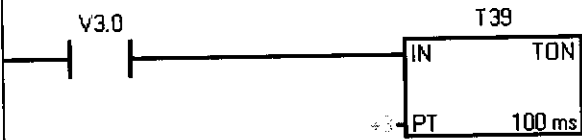
**Network 4**



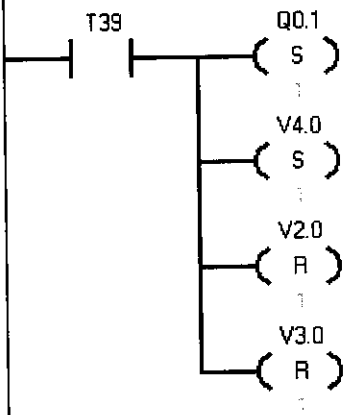
**Network 5**



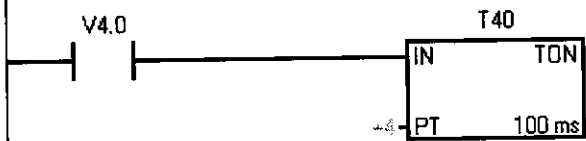
**Network 6**



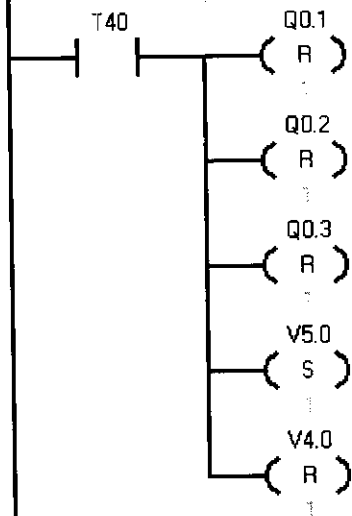
**Network 7**



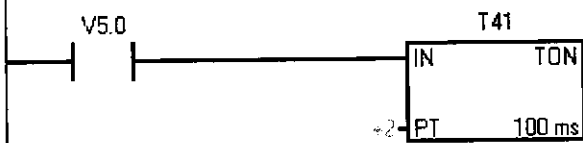
**Network 8**



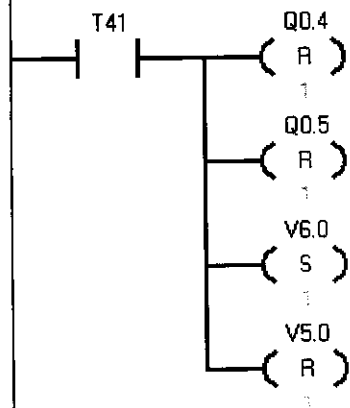
**Network 9**



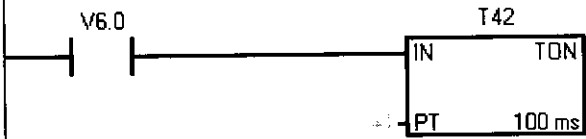
**Network 10**



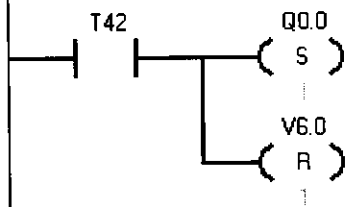
**Network 11**



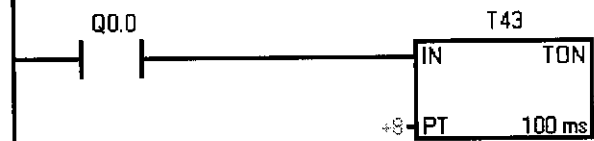
**Network 12**



**Network 13**

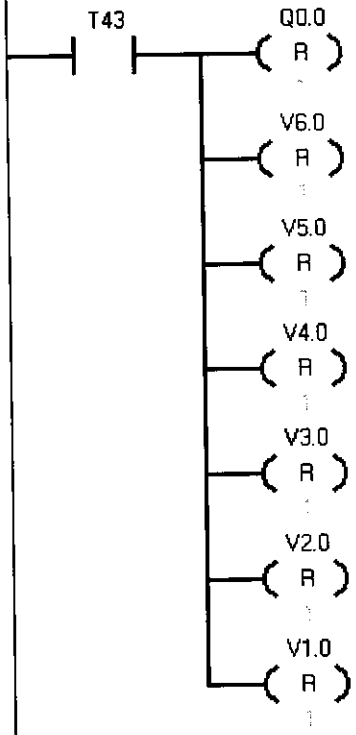


**Network 14**





Network 15



## **3.5. PLC TOOLS**

### **3.5.1 STEP 7-Micro/WIN Software**

STEP 7-Micro/WIN software is used for the full benefit of all the advantages of SIMATIC S7-200. Exceptionally powerful tools make daily work easier and save time and money. The intuitive wizards simplify the configuration of communications, and the new Super tree provides an optimized project overview as well as fast access to all segments of the user program. The interactive control panels for positioning as well as PID Auto tuning support the commissioning process even more efficiently than in the past.

The various wizards of STEP 7-Micro/WIN enable to use convenient Parametering in the place of programming. All the steps in the process are made a lot simpler: First enter a few parameters – aided by menus – to generate an executable program at the touch of a few keys.

#### **3.5.1.1 Download and installation instructions:**

1. To download the related files click the filename, then click "save". The files can be saved in a directory of choice. Repeat the process and save the files in the same directory.
2. As soon as all files are saved in the directory, click the "rejoin. bat" file. This causes the test version to be consolidated into a single .exe file.
3. To proceed with the installation, double-click the .exe file. This saves all Installs files in a temporary subdirectory on your PC and starts the installation.
4. Follow the instructions of the installation program.
5. When the installation is complete, the temporary subdirectory with the installation routines can be deleted. This action must be performed manually.

## **3.5.2 FEATURES**

### **3.5.2.1 Time Saving Data Entry**

- Freely-selectable programming using any editor: STL, LAD or FBD
- Trouble-free switching between STL, LAD or FBD at any time
- Convenient subroutine programming feature saves time and adds structure

Faster, better and more convenient with wizards

- Automatic program code generation, tailored to your application
- Simple, fast parameterization instead of time-consuming programming
- Wizards for PID control, CPU-to- CPU communication, high-speed counters, text display, position control and modem communication.

Everything self-explanatory

- Convenient user interface with Windows look and feel
- Fast and simple data entry using drag and drop
- Excellent visualization of projects and tools
- Comfortable documentation with context-sensitive online help

### **3.5.2.2 Password Protection**

Libraries and subroutines/functions can be password protected to safeguard your coding investments

### **3.5.2.3 Uniform Basic Functionality**

Uniform basic functionality across all editors with expansion for the STL editor, such as the symbol information table, drag and drop, network comments, etc.

### **3.5.2.4 Project Documentation**

Improved project documentation with print out options for the network wrapping, system block, local variables, cross references, etc.

**CHAPTER-4**  
**EMBEDDED SYSTEM**

## **4. AUTOMATION USING EMBEDDED SYSTEM**

### **4.1 Introduction**

The embedding of microprocessors into equipment and consumer appliances started before the appearance of the Personal Computers and consumes the majority of microprocessors. In this way, embedded microprocessors are more deeply ingrained into everyday life than any other electronic circuit that is made. A larger car may have over 50 microprocessors controlling functions such as the engine management systems, brakes with electronic anti-lock brakes, transmission with traction control and electronically controlled gearboxes, safety with airbag systems, electric windows, air-conditioning and so on.

An embedded system is a microprocessor-based system that is built to control a function or range of functions and is not designed to be programmed by the end user in the same way as a PC. An embedded system is designed to perform one particular task albeit with choices and different options. It is important because it differentiates itself from the world of the PC where the end user does reprogram it whenever a different software package is bought and run. However, PCs have provided an easily accessible source of hardware and software for embedded systems and it should be no surprise that they form the basis of many embedded systems.

## 4.2 STRUCTURE OF EMBEDDED SYSTEM

### Processor

The main criteria for the processor are: It provides the processing power needed to perform the tasks within the system. This seems obvious but it frequently occurs that the tasks are either underestimated in terms of their size and/or complexity or that creeping elegance expands the specification to beyond the processor's capability.

### Memory

Memory is an important part of any embedded system design and is heavily influenced by the software design, and in turn may dictate how the software is designed, written and developed. Memory performs two important functions within an embedded system:

- It provides storage for the software that it will run.

At a minimum, this will take the form of some non-volatile memory that retains its contents when power is removed. This can be on-chip read only memory (ROM) or external EPROM. The software that it contains might be the complete program or an initialization routine that obtains the full software from another source within or outside of the system. This initialization routine is often referred to as a bootstrap program or routine.

- It provides storage for data such as program variables and intermediate results, status information and any other data that might be created throughout the operation.

Software needs some memory to store variables and to manage software structures such as stacks. The amount of memory that is needed for variables is frequently less than that needed for the actual program. With RAM being more expensive than ROM and non-volatile, many embedded systems and in particular, micro controllers, have small amounts of RAM compared to ROM that is available for the program.

## **Peripherals**

An embedded system has to communicate with the outside world and this is done by peripherals. Input peripherals are usually associated with sensors that measure the external environment and thus effectively control the output operations that the embedded system performs.

The main types of peripherals that are used include:

- **Binary Outputs**

These are simple external pins whose logic state can be controlled by the processor to either be a logic zero (off) or logic one (on). They can be used individually or grouped together to create parallel ports where a group of bits can be input or output simultaneously.

- **Serial outputs**

These are interfaces that send or receive data using one or two pins in a serial mode. They are less complex to connect but are complicated to program. A serial port has to have data loaded into a register and then a start command issued.

- **Analogue values**

While processors operate in the digital domain, the natural world does not and tends to orientate to analogue values. As a result, interfaces between the system and the external environment need to be converted from analogue to digital and vice versa.

- **Displays**

Displays are becoming important and can vary from simple LEDs and seven segment displays to small alphanumeric LCD panels.

- **Time derived outputs**

Timers and counters are probably the most commonly used functions within an embedded system.

## Software

The software component within an embedded system often encompasses the technology that adds values to the system and defines what it does and how well it does it. The software can consist of several different components:

- Initialization and configuration
- Operating system or run-time environment
- The applications software itself
- Error handling
- Debug and maintenance support

## 4.3 FEATURES

### 4.3.1 Micro controller Core Features:

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches, which are two cycle
- Operating speed: DC - 20 MHz clock input DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,  
Up to 368 x 8 bytes of Data Memory (RAM)  
Up to 256 x 8 bytes of EEPROM data memory
- Pin out compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable  
Operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS FLASH/EEPROM technology



- Fully static design
- In-Circuit Serial Programming (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption

#### **4.3.2 Peripheral Features:**

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler can be incremented during sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. Resolution is 12.5 ns
  - Compare is 16-bit, max. Resolution is 200 ns
  - PWM max. Resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI (Master Mode) and I2Cä (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
  - Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
  - Brownout detection circuitry for Brown-out Reset (BOR)

### **4.3.3 Embedded system features**

#### **Provides easy maintenance**

The same mechanism that allows new functionality to be added through reprogramming is also beneficial in allowing bugs to be solved through changing software. Again it can reduce the need for expensive repairs and modifications to the hardware.

#### **Improves mechanical performance**

For any electromechanical system, the ability to offer a finer degree of control is important. It can prevent excessive mechanical wear, better control and diagnostics and, in some cases, actually compensate for mechanical wear and tear.

## **4.4 MICRO CONTROLLER**

Micro controller can be considered as self-contained systems with a processor, memory and peripherals so that in many cases all that is needed to use them within an embedded system are to add software. The processors are usually based on 8 bit stack-based architectures such as the MC6800 family. These are limited in their functionality but their low cost has meant that they are used in many obscure applications. Micro controllers are available in several forms:

- Devices for prototyping or low volume production runs
- Devices for low to medium volume production

### **4.4.1 Parallel ports**

Parallel ports provide the ability to input or output binary data with a single bit allocated to each pin within the port. They are called parallel ports because the initial chips that provided this support grouped several pins together to create a controllable data port similar to that used for data and address buses. It transfers multiple bits of information simultaneously, hence the name parallel port. Although the name implies that the pins are grouped together, the individual bits and pins within the port can usually be used independently of each other.

the pins are grouped together, the individual bits and pins within the port can usually be used independently of each other.

#### **4.4.2 Pull-up resistors**

It is important to check if a parallel port I/O port or pin expects an external pull-up resistor. Some devices incorporate it internally and therefore do not need it. If it is needed and not supplied, it can cause incorrect data on reading the port and prevent the port from turning off an external device.

#### **4.4.3 Timer/counters**

Digital timer/counters are used throughout embedded designs to provide a series of time or count related events within the system with the minimum of processor and software over-head. Most embedded systems have a time component within them such as timing references for control sequences, to provide system ticks for operating systems and even the generation of waveforms for serial port baud rate generation and audible tones.

#### **4.4.4 Downloading**

There are several methods available to get the code down to the target board;

- Serial lines

Programs can be downloaded into a target board using a serial comms port and usually an onboard debugger. The first stage is to convert the executable program file into an ASCII format which can be easily be transmitted to the target. This is done either by setting a switch on the linker to generate a download format or by using separate utility.

Several formats exist for the download format, depending on which processor family is being used.

The host is then connected to the target, invokes the download command on the target debugger and sends the file. The target debugger then converts the ASCII format back into binary and loads at the correct location. Once complete, the target debugger can be used to start the program.

- EPROM and flash

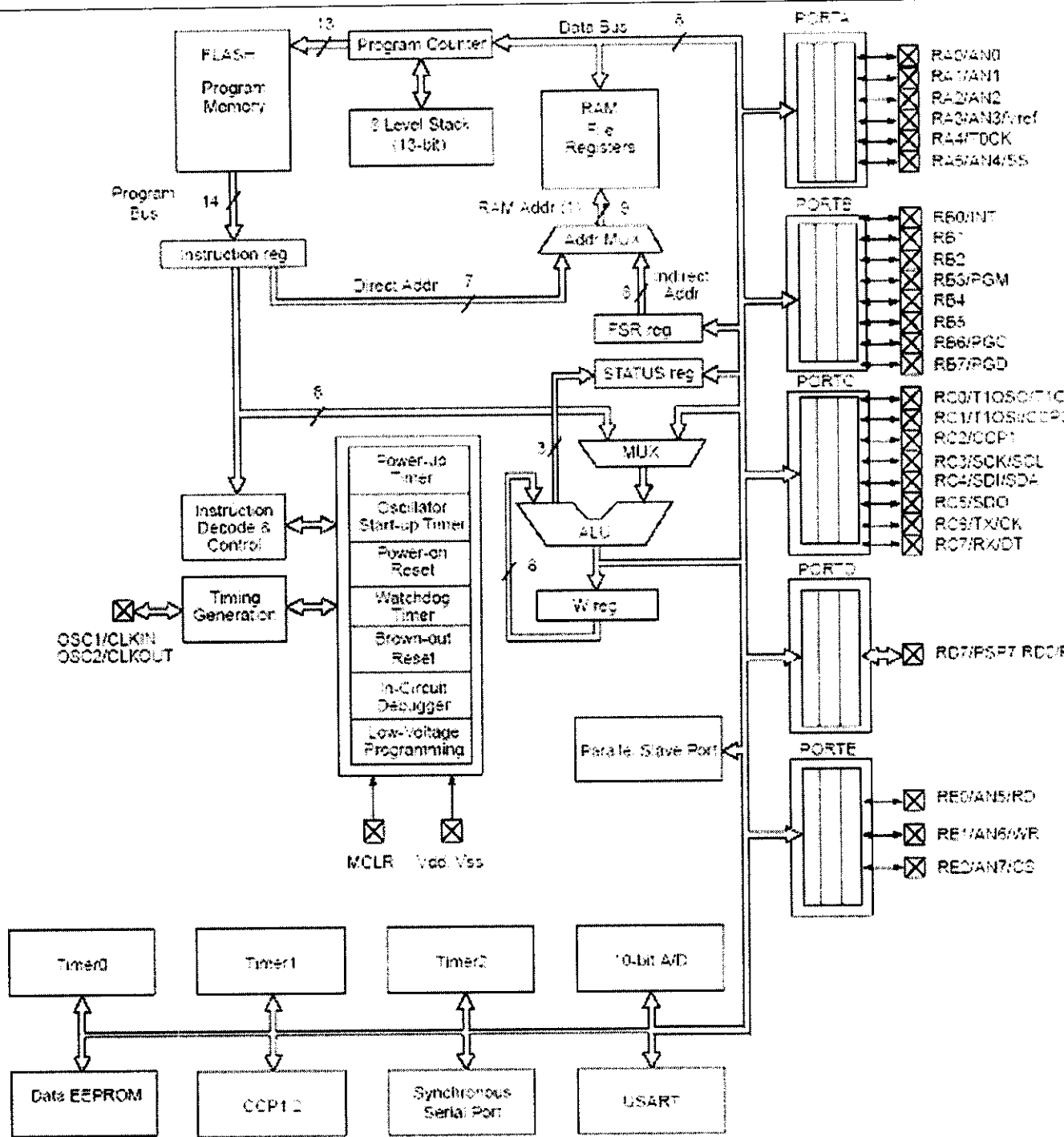
An alternative is to burn the program into EPROM, or some other form of non volatile memory such as FLASH or battery backed-up SRAM, insert the memory chips into the target and start running the code. This can be a lot quicker than downloading via serial line provided the link between the development system and the PROM programmer is not a serial link itself.

- Parallel ports

This is similar to the serial line technique, except that data is transferred in bytes rather than bits using a parallel interface – often a reprogrammed Centronics printer port. While a lot faster, it does require access to parallel ports which tend to be less common than serial lines.

#### **4.4.5 Debugging techniques**

The fundamental aim of a debugging methodology is to restrict the introduction of untested software or hardware to a single item. This is good practice and of benefit to the design and implementation of any system. It is to prevent this integration of two unknowns that simulation programs to simulate and test software, hardware or both can play a critical part in the development process.



Note 1: Higher order bits are from the STATUS register

Fig 1.6: Architecture of PIC 16F877A

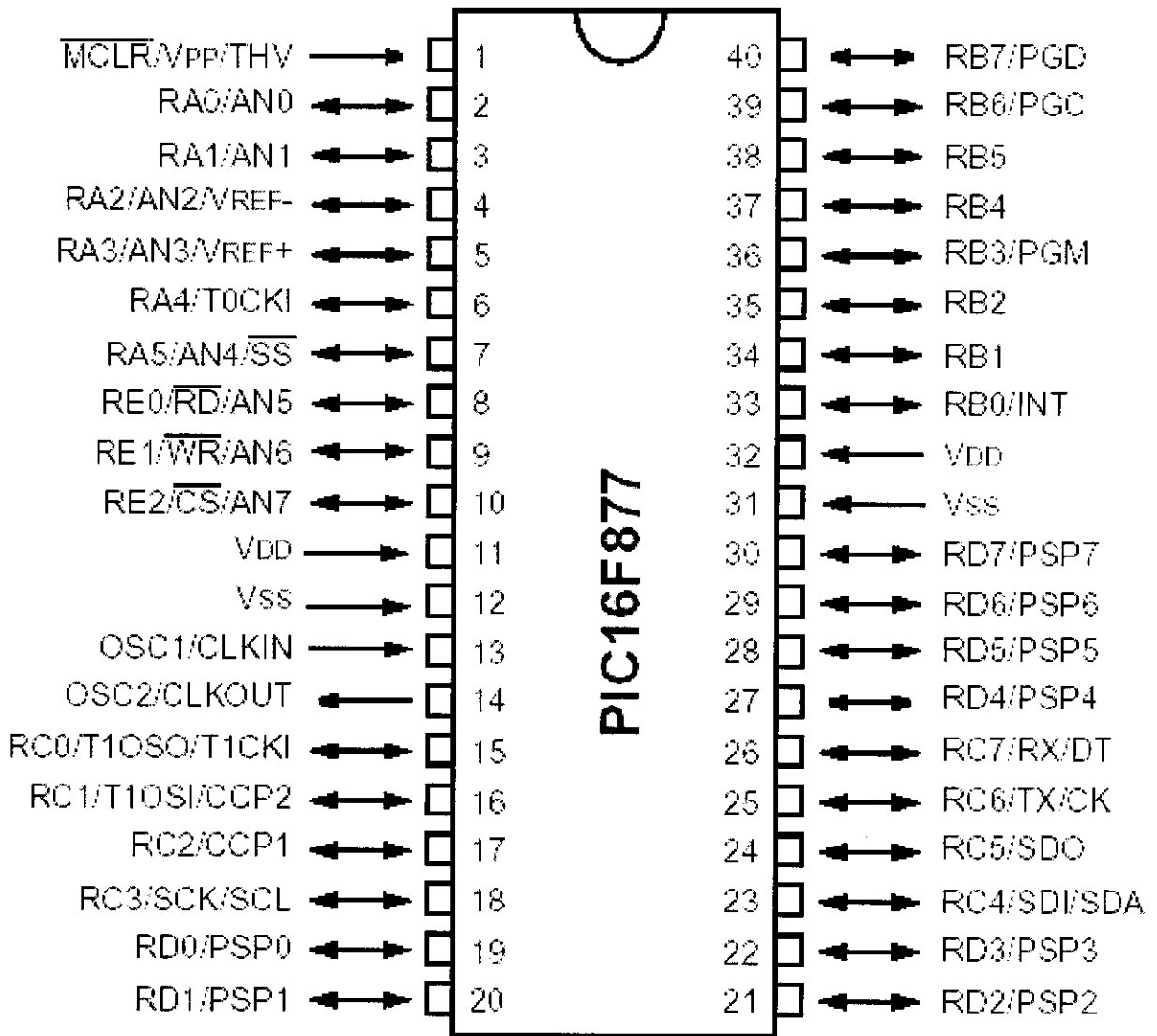


Fig 1.7: PIN DIAGRAM OF PIC 16F877A

## Input output switches

INPUT	PARTS	SWITCHES
Port C 3 <sup>rd</sup> pin	Carriage Forward Limit Switch	LS1
Port C 4 <sup>th</sup> pin	Push Button_1 Limit Switch	LS2
Port C 5 <sup>th</sup> pin	Push Button_2 Limit Switch Assembly	LS3
Port C 0 <sup>th</sup> pin	Assembly Forward Limit Switch	LS4
Port C 1 <sup>st</sup> pin	Pre- Assembly Forward Limit Switch	LS5
Port C 2 <sup>nd</sup> pin	Post- Assembly Forward Limit Switch	LS6

Table 1.4: Embedded Input

OUTPUT	PARTS	SWITCHES
Port B 4 <sup>th</sup> pin	Assembly Cylinder Solenoid Valve	SV1
Port B 2 <sup>nd</sup> pin	Pre-press Solenoid Valve	SV2
Port B 1 <sup>st</sup> pin	Post- press Solenoid Valve	SV3
Port B 0 <sup>th</sup> pin	Holding- Cylinder Solenoid Valve	SV4
Port B 3 <sup>rd</sup> pin	Carriage Ejector Solenoid Valve	SV5

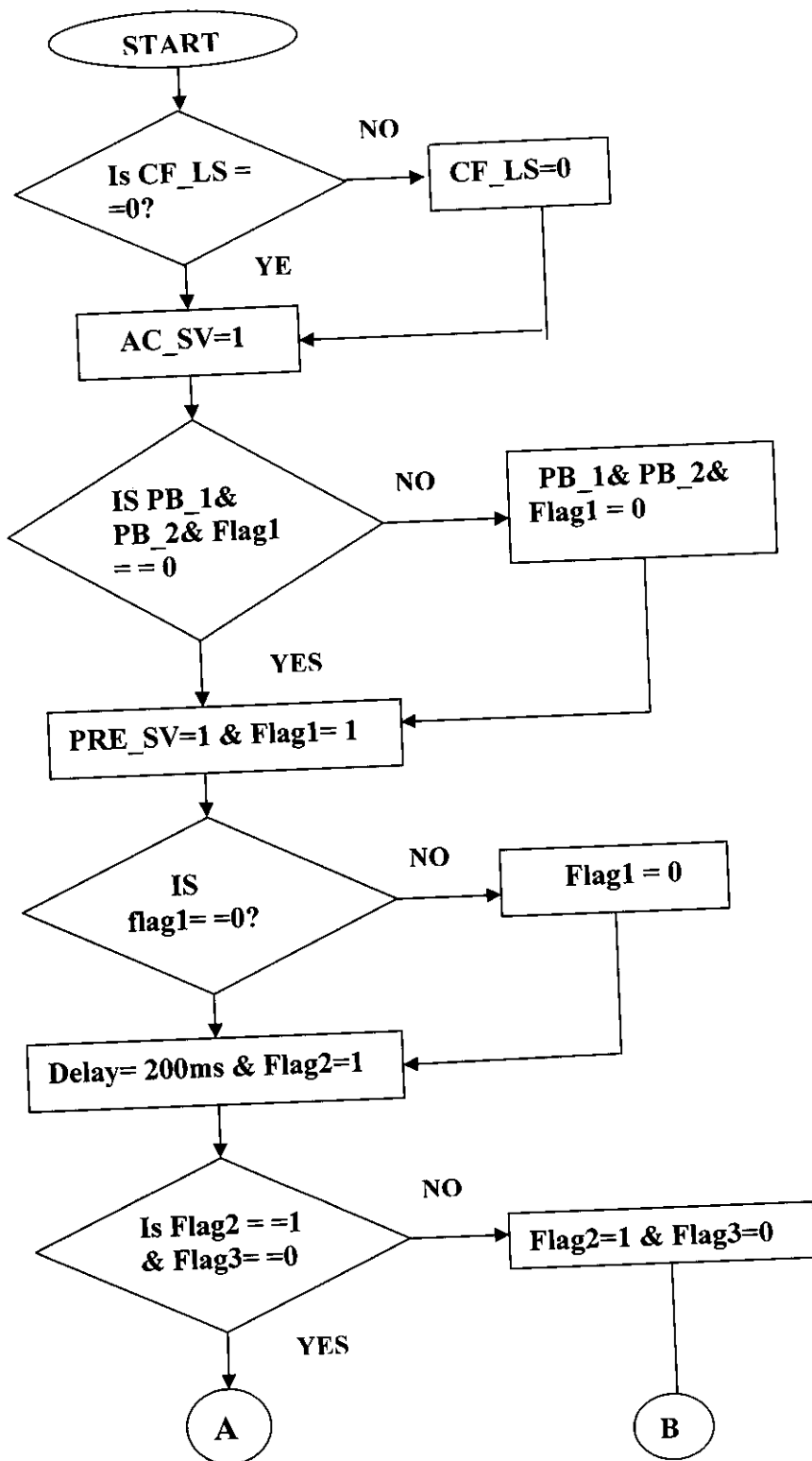
Table 1.5: Embedded output

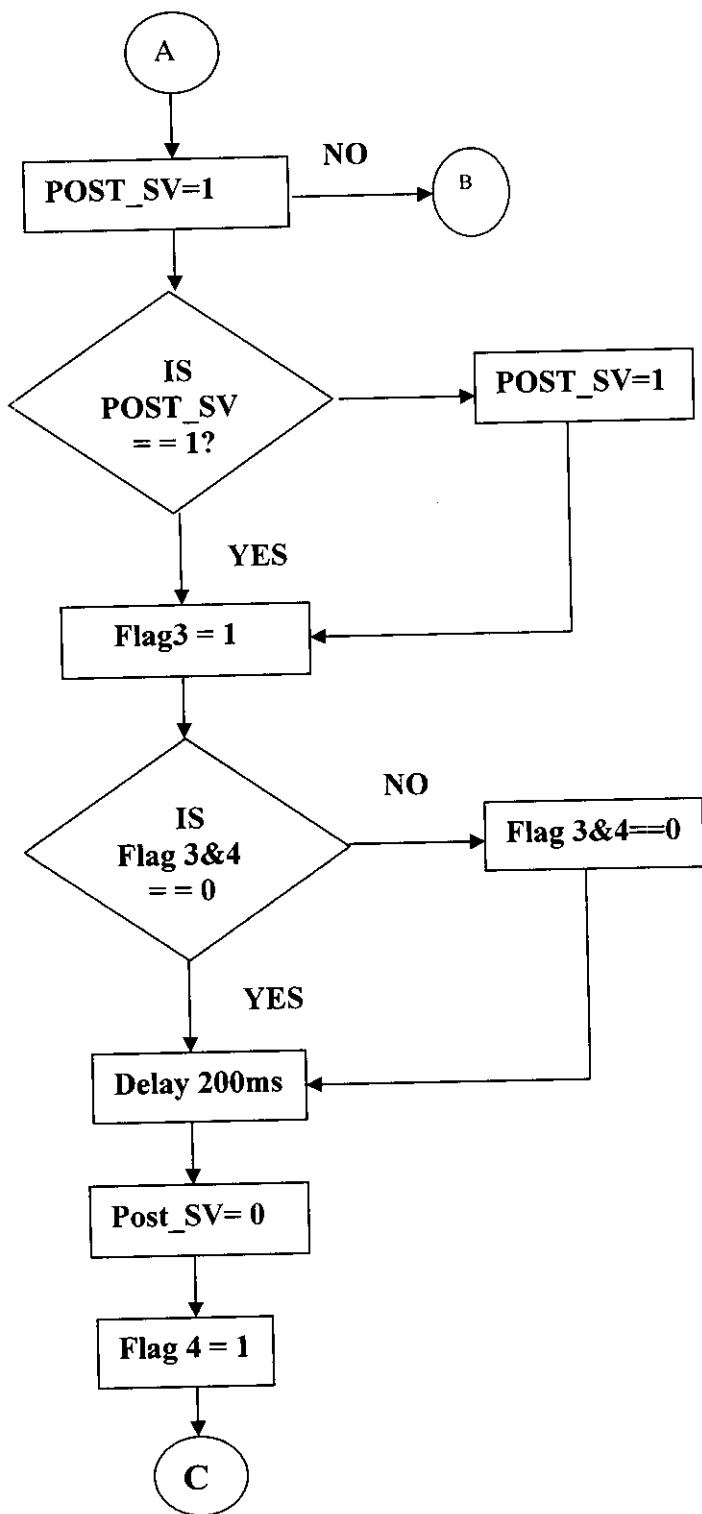


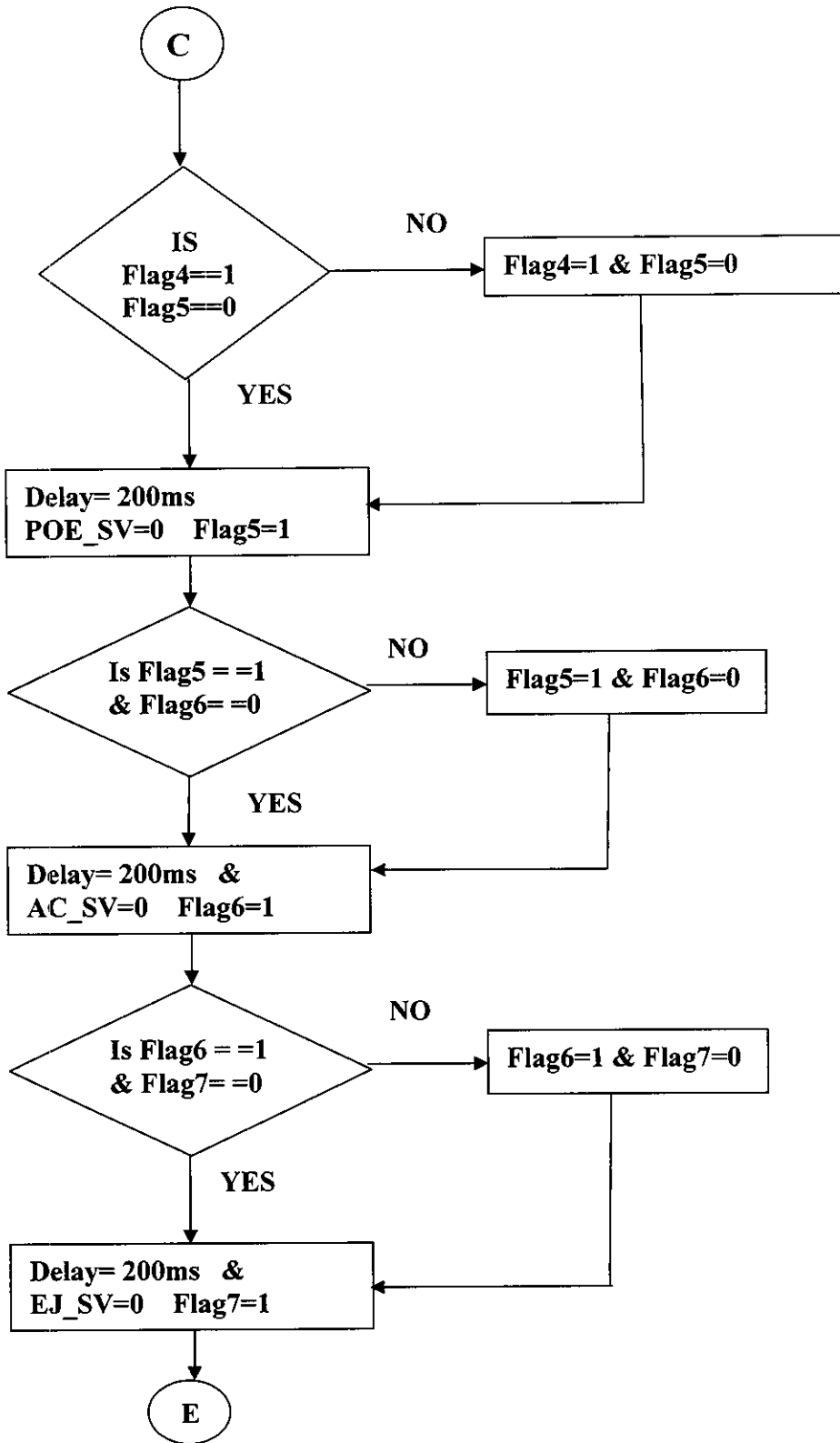


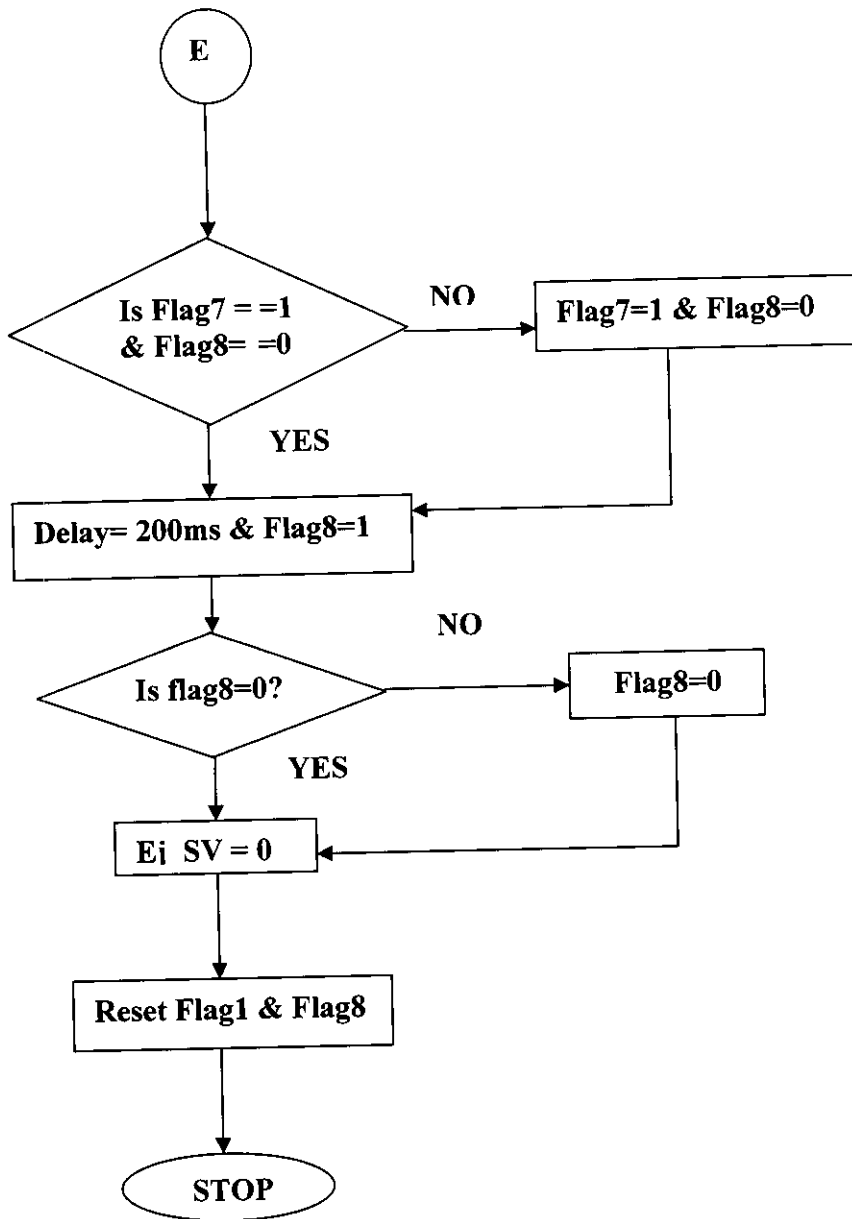


**Fig1.8. Flow chart for embedded process**









## 4.5 EMBEDDED PROGRAM

```
// *****CONTACT FIXING MACHINE*****//
#include "C:\contact fixing\contact fixing.h"

// INPUT CONFIG

#bit CF_LS=0x07.3
#bit PB_1=0x07.4
#bit PB_2=0x07.5
#bit AF_LS=0x07.0
#bit PRE_LS=0x07.1
#bit POST_LS=0x07.2

//OUTPUT CONFG

#bit AC_SV=0x06.4
#bit PAC_SV=0x06.2
#bit POAC_SV=0x06.1
#bit HO_SV=0x06.0
#bit EJ_SV=0x06.3

// REG CONFIG

#byte trisc=0x87
#byte trisb=0x86
#byte trisa=0x85
#byte portb=0x06
int flag1,flag2,flag3,flag4,flag5,flag6,flag7,flag8,flag9,flag10=0;

void main()

{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_CLOCK_DIV_2);
  setup_spi(FALSE);
  setup_psp(PSP_DISABLED);
  setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_ccp1(CCP_OFF);
  setup_ccp2(CCP_OFF);
  trisc=0xff;
  trisb=0;
  portb=0;
}
```

```
flag1,flag2,flag3,flag4,flag5,flag6,flag7,flag8,flag9,flag10=0;
```

```
while(1)
```

```
{
```

```
if (cf_ls==0)
```

```
{
```

```
Ac_sv=1;
```

```
}
```

```
if ((pb_1==0)&&(pb_2==0)&&(flag1==0))
```

```
{
```

```
pac_sv=1;  
flag1=1;
```

```
}
```

```
if (flag1==1)
```

```
{
```

```
Delay_ms(200);  
Flag2=1;
```

```
}
```

```
If ((flag2==1) &&(flag3==0))
```

```
{
```

```
Poac_sv=1;
```

```
}
```

```
if ( Poac_sv==1)
```

```
{
```

```
flag3=1;
```

```
}
```

```
If ((flag3==1) &&(flag4==0))  
    {
```

```
        Delay_ms(200);  
        Poac_sv=0;  
        Flag4=1;
```

```
    }  
If ((flag4==1) &&(flag5==0))
```

```
    {
```

```
        Delay_ms(200);  
        Pac_sv=0;  
        flag5=1;
```

```
    }
```

```
If ((flag5==1) &&(flag6==0))
```

```
    {
```

```
        Delay_ms(200);  
        Ac_sv=0;  
        flag6=1;
```

```
    }
```

```
If ((flag6==1) &&(flag7==0))
```

```
    {
```

```
        Delay_ms(200);  
        Ej_sv=1;  
        flag7=1;
```

```
    }
```

```
If ((flag7==1) &&(flag8==0))
```

```
    {
```

```
        Delay_ms(200);  
        flag8=1;
```



```
    }  
    If (flag8==1)  
    {  
        Ej_sv=0;  
        flag1=flag2=flag3=flag4==0;  
    }
```

```
}
```

```
}
```

```
// END OF THE PROG
```



P-2366

## 4.5.1 CODE SYNTAX

**#bit CF\_LS=0x07.3**

**Carriage Forward limit switch Configured to Port C 3<sup>rd</sup> pin.**

**#bit PB\_1=0x07.4**

**Left push button limit switch Configured to Port C 4<sup>th</sup> pin.**

**#bit PB\_2=0x07.5**

**Right push button limit switch Configured to Port C 5<sup>th</sup> pin.**

**#bit AF\_LS=0x07.0**

**Assembly forward limit switch Configured to Port C 0<sup>th</sup> pin.**

**#bit PRE\_LS=0x07.1**

**Pre press limit switch Configured to Port C 1<sup>st</sup> pin.**

**#bit POST\_LS=0x07.2**

**Post press limit switch Configured to Port C 2<sup>nd</sup> pin.**

**#bit AC\_SV=0x06.4**

**Assembly cylinder Solenoid Valve Configured to Port B 4<sup>th</sup> PIN**

**#bit PRE\_SV=0x06.2**

**Pre press Solenoid Valve Configured to Port B 2<sup>nd</sup> PIN**

**#bit POST\_SV=0x06.1**

**Post press Solenoid Valve Configured to Port B 1<sup>st</sup> PIN**

**#bit HO\_SV=0x06.0**

**Holding Solenoid Valve Configured to Port B 0<sup>th</sup> PIN**

**#bit EJ\_SV=0x06.3**

**Carriage ejector Solenoid Valve Configured to Port B 3<sup>rd</sup> PIN**

**#byte trisc=0x87**

**Port C assigned as input.**

**#byte trisb=0x86**

**Port b assigned as input**

**#byte trisa=0x85**

**Port a assigned as input**

**trisc=0xff;**

**Port C assigned as Input**

**trisb=0;**

**Port b assigned as Output**

**portb=0;**

**Clear Port B**

## 4.6. PIC TOOLS

### 4.6.1 C COMPILER

The compiler contains Standard C operators and built-in libraries that are specific to the PIC registers. Access to hardware features from C.

#### 4.6.1.1 Features includes:

- 1, 8, 16 and 32 bit integer types and 32 bit floating point.
- Standard one bit type (Short Int) permits the compiler to generate very efficient Bit oriented code.
- #BIT and #BYTE will allow C variables to be placed at absolute addresses to map registers to C variables.
- Bit Arrays
- Fixed Point Decimal
- Constants (including strings and arrays) are saved in program memory.
- Flexible Handling Of Constant Data
- Variable Length Constant Strings
- Addressor Capability To Create User Defined Address Spaces In Memory Device

The CCS C Compiler for PIC10, PIC12, PIC14, PIC16, and PIC18 microcontrollers has over 180 Built-in Functions to access PIC<sup>®</sup> MCU hardware is easy and produces efficient and highly optimized code. Functions such as timers, A/D, EEPROM, SSP, PSP, USB, I<sup>2</sup>C and more:

- Built-in libraries that work with all chips for RS-232 serial I/O, I2C, discrete I/O and precision delays.
- Serial I/O functions allow standard functions such as GETC () and PRINTF () to be used for RS-232 like I/O.
- Formatted printf allows easy formatting and display in HEX or decimal.
- Multiple I2C and RS232 ports may be easily defined.

- #Use rs232 () offers options to specify a maximum wait time forgetc.
- Hardware transceiver used when possible, but for all other occasions the compiler generates a software serial transceiver.
- Microcontroller clock speed may be specified in a PRAGMA to permit built-in functions to delay for a given number of microseconds or milliseconds.
- Functions such as INPUT () and OUTPUT\_HIGH () properly maintain the tri-state registers.
- Compiler directives determine if tri-state registers are refreshed on every I/O or if the I/O is as fast as possible.
- #USE SPI ()
- Simple functions like READ\_ADC () to read a value from A/D converter.
- Source code drivers included for LCD modules, keypads, 24xx and 94xx serial EEPROM, X10, DS1302 and NJU6355 real time clocks, Dallas touch memory devices, DS2223 and PCF8570, LTC1298 and PCF8591 A/D converters, temperature sensors, digital pots, I/O expander and much more.

The compiler can handle inline or separate functions, as well as parameter passing in re-usable registers. Transparent to the user, the compiler handles calls across pages automatically and analyzes program structure and call tree processes to optimize RAM and ROM Usage.

#### **4.6.1.2 Additional features**

Efficient function implementation allows call trees deeper than the hardware stack.

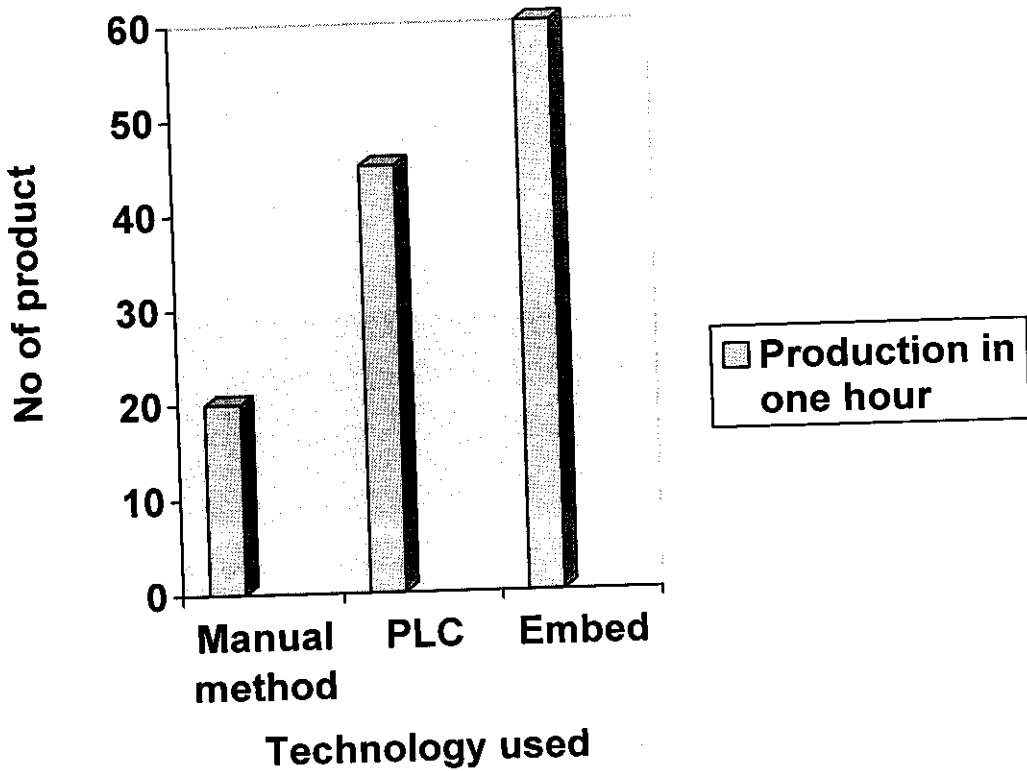
- Automatic linking handles multiple code pages.
- Assembly code may be inserted anywhere in the source and may reference C variables.
- Function Overloading allows for several functions with the same name, but differences in number and type of parameters.

- Default Parameters can be used in a function if arguments are not used in a call.
- Interrupt functions supported on PCM/PCH. The compiler generates all startup and clean up code as well as identifying the correct interrupts function to be called.
- Reference parameters may be used to improve code readability and inline function efficiency.
- Generation Of Multiple HEX Files For Chips With External Memory
- Variable Number Of Parameters in a function.
- Relocatable Objects / Multiple Compilation Unit (IDE Only)
- Automatic #fuses Configuration

The compiler runs under Windows 95, 98, ME, NT4, 2000, XP, Vista, or Linux. It outputs hex and debug files that are selectable and compatible with popular emulators and programmers including the MPLAB® IDE for source level debugging.

- Updates via the Internet for 30 days included.
- CCS Backwards Compatibility

**CHAPTER-5**  
**COMPARISON**  
**RESULT**



**Graph5.1: Comparison result**

## **RESULT**

Thus the graph shows the gradual increase in the production of the contactor by using PLC (S7-226) and PIC micro controller (16F877A). The existing method results in the production of 20 pieces per hour. PLC gives the production of about 45 pieces per hour and PIC micro controller increases the production to 60 pieces per hour.



**CHAPTER-6**  
**CONCLUSION**  
**&**  
**RECOMMENDATION**

## **6. CONCLUSIONS AND FUTURE SCOPE**

### **6.1 CONCLUSION FROM THE PROJECT**

This project is based on an innovative idea. The automation has done using programmable logic controller and peripheral interface controller in order to insert the contacts in the contactor. Thus the automation process helps in increasing the total production, profit and in saving time.

Since the project is integrated with automation of contacts insertion, it will minimize the human errors as the human labour is reduced.

## 6.2 FUTURE SCOPE

1. The contacts that are placed manually in the contactor now, can also be placed without manual interruption using vibrator bowl in which the contacts get automatically placed in the contactor under certain vibration.
2. Next processes such as ultrasonic welding and testing the contactor can be combined together with the insertion process and can be automated in a single machine itself. Thereby time for whole contactor assembly and testing could be reduced more and productivity can be further increased.

# **BIBLIOGRAPHY**

## **BIBLIOGRAPHY**

### Books

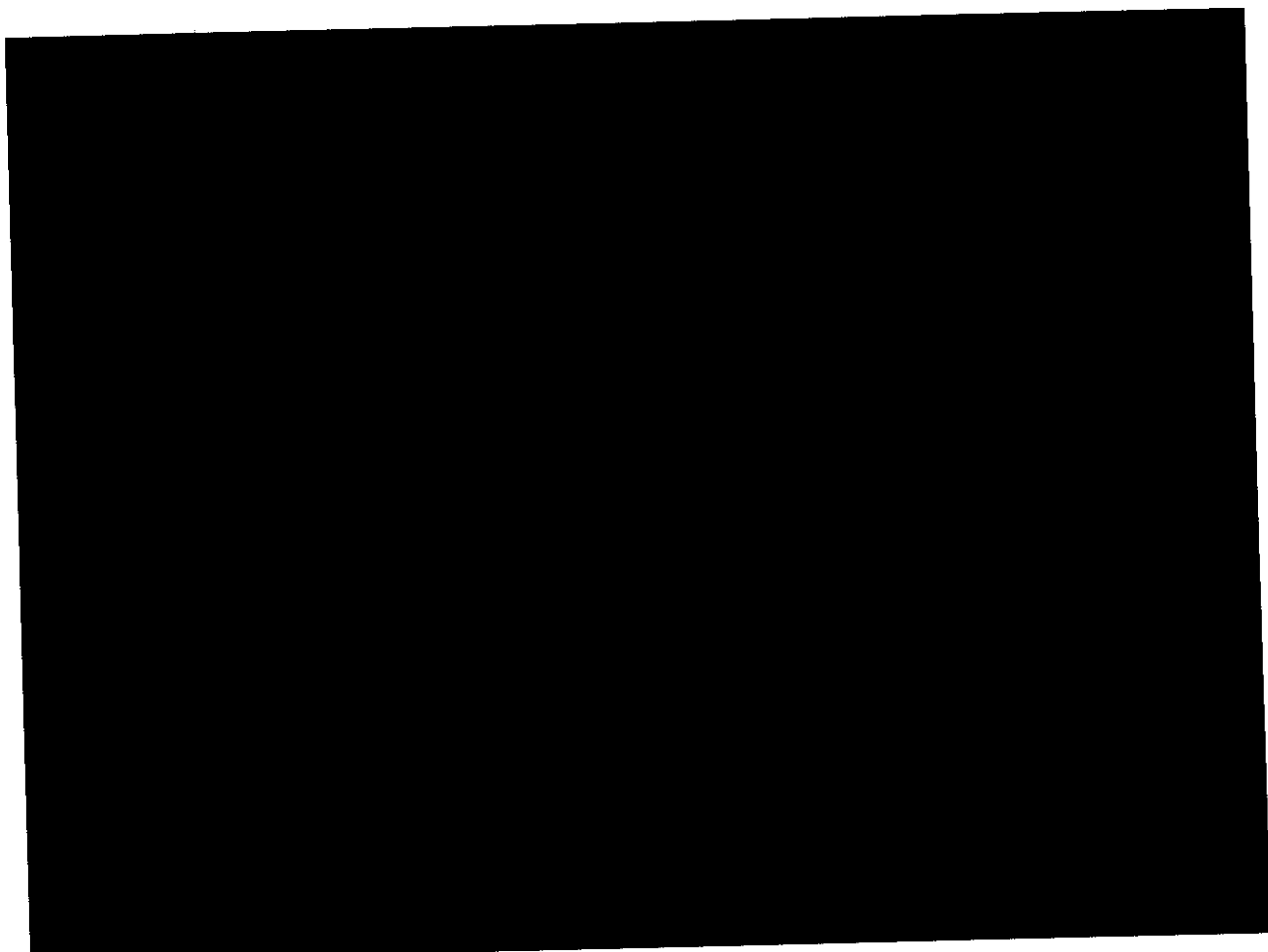
1. Steve Heath, 'Design of Embedded System' 3<sup>rd</sup> edition, 2004.
2. Michael Barr, 'Programming Embedded System in C & C++', 2001.
3. Richard H.Barnett, 'Embedded C Programming & the Microchip PIC, 2005.
4. Frank D. Petruzella, 'Programmable logic controllers', 2000.

### Web sites

1. [www.wikipedia.com](http://www.wikipedia.com)
2. [www.microchip.com](http://www.microchip.com)
3. [www.datasheets4u.com](http://www.datasheets4u.com)



## Machine Diagram





Inputs to PLC





PCB for Embedded System