Palestine Polytechnic University

College of Engineering & Technology

Electrical & Computer Engineering Department

Graduation Project

Replacing PLC controller with PIC

Case: Packaging machine of sugar bags

Project team

Sami Rateb Badr

Mahmmoud Ahmad Abu Amriah

Project Supervisor

Eng. Mazen Zalloum

Hebron – Palestine

2011

جامعة بوليتكنك فلسطين

الخليل - فلسطين

كلية الهندسة و التكنولوجيا

دائرة الهندسة الكهربائية و الحاسوب

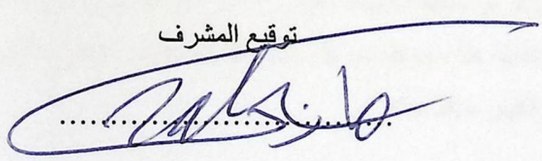
Replacing PLC controller by PIC

Case: Packaging machine of sugar bags

سامي راتب بدر **محمود أحمد أبو عمريه**

بناء على نظام كلية الهندسة و التكنولوجيا و إشراف و متابعة المشرف المباشر على المشروع و موافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دارة الهندسة و الكهربائية و الحاسوب و ذلك للوفاء بمتطلبات درجة الباكالوريوس في الهندسة تخصص أنظمة الحاسوب.

توقيع المشرف



.....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

.....

Palestine Polytechnic University

PPU

Replacing PLC controller with PIC

Case: Packaging machine of sugar bags

By

Sami Rateb Badr Mahmmoud Ahmad Abu Amriah

Chairperson of the Supervisor Committee: Eng. **Mazen Zalloum**

Department of Electrical and computer Engineering

ABSTRACT

هذا المشروع يهدف إلى استبدال المتحكم PLC بمتحكم من نوع PIC و ذلك من أجل توفير حل هندسي يقوم على توفير المال من أجل الوصول إلى الهدف المنشود ، و الهدف تصميم تطبيق صناعي يعمل على المتحكم PIC مع بعض دوائر التوصيل اللازمة ويستطيع أن يقوم بوظائف المتحكم PLC و ذلك عن طريق برمجته بلغة C البسيطة .

و التطبيق الصناعي الذي اخترناه هو ماكينة تغليف أكياس السكر الورقية بوزن واحد كيلو جرام ، و التي تحتوي على محرك ثلاث فاز بقدرة ١٢٠ وات ، وظيفة هذا محرك تحريك السير الناقل بطول ٢ متر و الذي يقوم بنقل كيس السكر المعبئ لإجراء عملية التغليف اللازمة ليخرج الكيس مغلفا جاهزا.

This project aims to replace a PLC microcontroller with a PIC microcontroller in order to find an engineering alternative solution for saving money, in order to achieve the purpose. The main target is to design an industrial application that is operated by PIC microcontroller and some necessary connection circuits, and to simulating the PLC with C program.

The application we chose is a packaging machine of sugar bags of one kilogram, which contain 3 phase motor with 120 watt to drive the conveyor (2 meter) which moves the filled bag to package.

الإهـــــــــــداء

إلى من يسعد قلبي بلقياها

إلى روضة الحب التي تنبت أزكى الأزهار ... **أمي**

إلى رمز الرجولة والتضحية

إلى من دفعني إلى العلم وبه ازداد افتخاراً ... **أبي**

إلى من هم اقرب ألي ّ من روحي

إلى من شاركني حضن ألام وبهم استمد عزتي وإصراري ... **إخوتي**

إلى من آنسني في دراستي

وشاركني همومي تذكاراً وتقديراً ... **أصدقائي**

إلى هذه الصرح العلمي المتميز الرائع

...

جامعة بوليتكنك فلسطين

كلمة شكر

لابد لنا ونحن نخطو خطواتنا الأخيرة في الحياة الجامعية من وقفة نعود إلى أعوام قضيناها في رحاب الجامعة مع أساتذتنا
الكرام الذين قدموا لنا الكثير باذلين بذلك جهودا كبيرة في بناء جيل الغد لتبعث الأمة من جديد...
وقبل أن نمضي تقدم أسمى آيات الشكر والامتنان والتقدير والمحبة إلى الذين حملوا رسالة العلم...
إلى الذين مهدوا لنا طريق العلم والمعرفة...
إلى جميع أساتذتنا الأفاضل.....

"كن عالما ... فإن لم تستطع فكن متعلما ، فإن لم تستطع فأحب العلماء ،فإن لم تستطع فلا تبغضهم"

و أخص بالشكر و التقدير مشرف المشروع :

المهندس مـــازن زلـــــوم

كما نتقدم بالشكر إلى كل من:

المهندسة هـــبـــة التميمي

المهندس ســـامي السلامين

المهندس كريـــــم الـجـنيدي

المهندس مكـــاوي حـريـــز

وإلى كل من مد يد العون لإنجاز هذا العمل.

Contents

List of Figures

List of Tables

CHAPTER ONE

INTRODUCTION

1.1 Introduction:

In the sixties and the seventies, industry has begun to recognize the need for quality improvement and productivity increase. Flexibility, that is the ability to change a process quickly in order to satisfy consumer needs, also became a major concern.

Increased competition forced system-designers to improve production quality and productivity. Flexibility, as well as fast and easy change, of production became crucial, so the designers suggested an idea which is to use system logic of the microcomputers instead of wired relays. If changes were needed in system logic, or in order of operations, the program in a microcomputer could be changed instead of rewiring the relays, considering only the limitation of time which is needed for changes in wiring to be done.

1.2 Project main Objectives:

The project has two basic objectives. The first is to design a PIC circuit that takes the place of the PLC, taking into account the environment and the difference between them, as a difference in the power and the current that supply each of them.

Another objective is to program the PIC to Simulate the PLC work, so that it could perform the same functions of the PLC, and work in the PLC application.

1.3 The main idea of the project:

The project aims to replace a PLC microcontroller with the PIC controller that is found in the "Packaging Machine of Sugar Bags" project.

1.4 The reasons for choosing this project:

- The cost of the PLC controller is very expensive; unlike this the PIC microcontroller is less expansive.(e.g. DELTA PLC is 150 $ and PIC18f4550 is 15 $)
- Market need such a machine we chosen in this project and widely common, but in less cost to be available.
- A PIC microcontroller is very versatile, so you can choose the best chip that you need. On the other hand, the PLC controller is not too versatile.
- PIC is possibly remote control and connects to internet.
- Connect between PIC and computer is more easer by using USB.

1.5 The Main parts of the project:

The project is divided into two important parts. The first one is the hardware phase and its elements, which includes interfacing PIC with the application and suitable ICs.

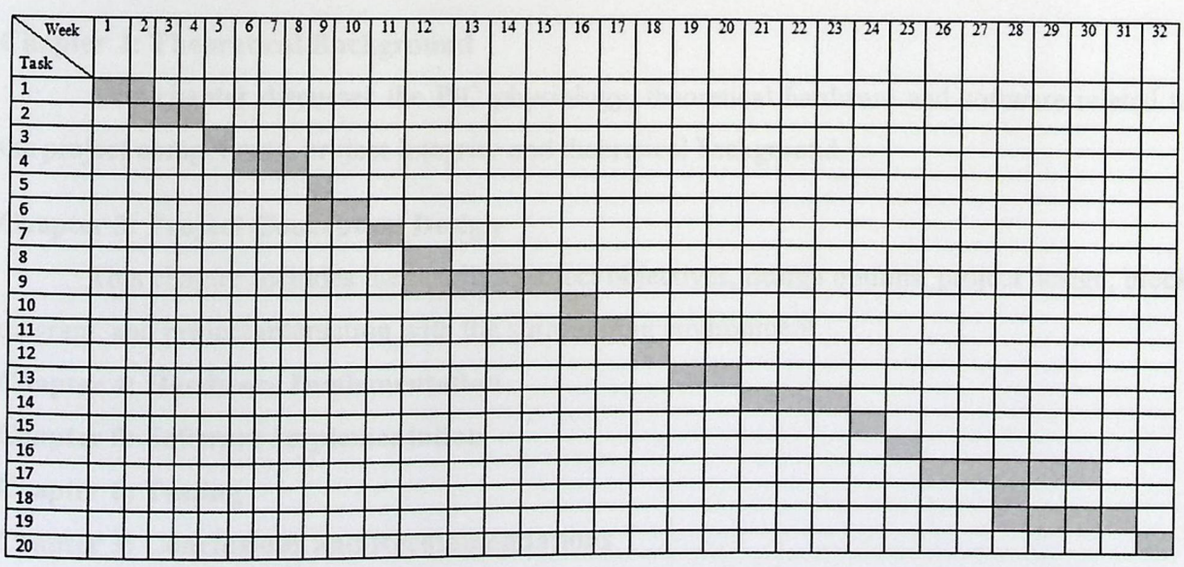
The second part is writing software in C#.net which makes a translation program of PLC to C language, in order to operate all of the hardware parts of this project.

1.6 Time Table:

Table 1-1: Time Schedule Table

Task#	Task	Duration(weeks)
1	Project determination	1 - 2
2	Getting ideas and important information about the project	2 - 4
3	Documentation chapter one	5
4	Documentation chapter two	6 - 8
5	Studding system requirements	9
6	Studding machine blocks diagrams	9 - 10
7	Project design and block	11
8	Documentation chapter three	12
9	Full documentation	12 - 15
10	Power point and prepare for presentation	16
11	ICs deterring and fetching	16 – 17
12	PIC interfacing	18
13	Relays and contactor connections	19 – 20
14	Software designing and building	21 – 23
15	Interfacing PIC with PC	24
16	Documentation chapter four	25
17	Testing the project	26 - 30
18	Documentation chapter five and six	27
19	Data analysis and conclusion with recommendation. Final documentation	28 – 31
20	Power point and prepare for presentation	32

Table 1-2: Time diagram



1.7 Estimated cost:

This section lists the overall cost of the project; the cost includes the hardware cost, the software cost.

Hardware cost: includes the costs of components that are to be used to implement the project. Table 1-3 shows these costs.

Table 1-3: Cost table

Component	Quantity	Pries ($)
PIC18F4550	1	10
ULN2804	1	3
Printed board	1	100
Effort	2 persons	500
PC	1	500
Visual studio software	1	100
Others	-	100
Total cost	-	1313

1.8 Report Road Map:

This report consists of four chapters; the following is a brief description of the topics that are covered in each chapter.

Chapter 1: Introduction

This chapter presents overview, literature review, project scheduling, and estimated cost, project risk.

Chapter 2: Theoretical Background

This chapter discusses the PIC physiology, theoretical hardware and software related to the project components, project integrity and theoretical background.

Chapter 3: Project Conceptual Design

This chapter includes the detailed project objectives, design options, project design, block diagram, and project interaction with the surrounding environment.

Chapter 4: Hardware Implementation

Chapter 5: Software Implementation

Chapter 6: Testing

Chapter 3: Conclusions and Recommendations

CHAPTER TWO

THEORETICAL BACKGROUN

2.1 Overview

2.2 Programmable Logic Controller

2.3 Packaging Machine of Sugar Bags: Design and Operating

2.4 PIC Vs PLC

2.5 PIC18F4550

2.6 Relay Connections

2.7 Limit Switch

2.8 Amplification Circuits

This chapter focuses on programmable logic controller PLC which we need to know about it, and packaging machine our application to replace PLC with PIC.

Programmable logic controller:

A programmable logic controller PLC or programmable controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures .PLCs are used in many industries and machines .Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact .

Programs to control machine operation are typically stored in battery-backed or non-volatile memory .

A PLC is an example of a real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.

- **Features:**

The mean features in PLC are:

1. **Input modules**: These connect the PLC to sensors and actuators , PLCs read limit switches, analog process variables such as temperature and pressure, modules can be :
   - Digital or Analog.
   - Number of modules 8 or 16.
   - The voltage is DC or AC.
   - The feed voltage is 24V or 120V or 240V.

2. **Output modules**: PLCs operate electric motors, pneumatic or hydraulic cylinders, magnetic relays, solenoids, and the modules can be :

2.1 Overview:

This chapter focuses on programmable logic controller PLC which we need to know about it, and packaging machine our application to replace PLC with PIC.

2.2 Programmable logic controller:

A programmable logic controller PLC or programmable controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures .PLCs are used in many industries and machines .Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact .

Programs to control machine operation are typically stored in battery-backed or non-volatile memory .

A PLC is an example of a real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.

- **Features:**

The mean features in PLC are:

1. **Input modules**: These connect the PLC to sensors and actuators , PLCs read limit switches, analog process variables such as temperature and pressure, modules can be :
   - Digital or Analog.
   - Number of modules 8 or 16.
   - The voltage is DC or AC.
   - The feed voltage is 24V or 120V or 240V.

2. **Output modules**: PLCs operate electric motors, pneumatic or hydraulic cylinders, magnetic relays, solenoids, and the modules can be :

2.1 Overview:

This chapter focuses on programmable logic controller PLC which we need to know about it, and packaging machine our application to replace PLC with PIC.

2.2 Programmable logic controller:

A programmable logic controller PLC or programmable controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures .PLCs are used in many industries and machines .Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact .

Programs to control machine operation are typically stored in battery-backed or non-volatile memory .

A PLC is an example of a real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.

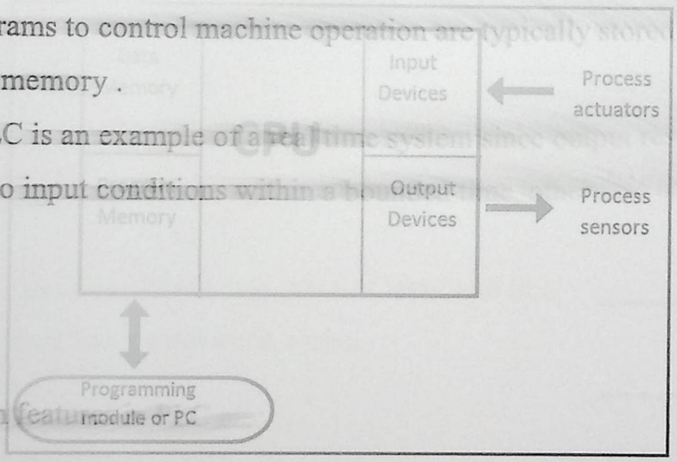


Figure 2.1: PLC components

- **Features:**

The mean features of PLC

1. **Input module:** These ... switches, analog ...
   - Digital or analog ...
   - Number ...

- Digital or Analog.
- Number of modules 8 or 16.
- The voltage is DC or AC.
- The feed voltage is 24V or 120V or 240V.
- The load current may 10mA, 1A, 2A, 10A and so on.

3. **The program Memory**: The instructions for logical control sequence are stored here.

4. **Data memory:** The status of switches, interlocks, past values of data and other working data is stored here .
5. **CPU:** Control processing unit.

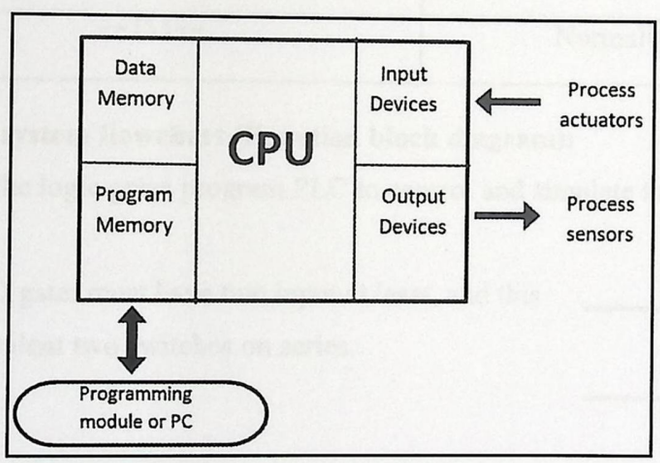


Figure 2.1: PLC components

- **Software :**

Software is a collection of commands to be logically executed, in order to control the applications.

- **Programming methods :**

To program PLC there is three methods:

1. Ladder diagram method: it's like drawing the circuit by using special icons, in this method the circuit drown horizontally, and this is the common way to programming PLC.

Table 2-1: Ladder symbols

Icon	Function
--( )--	Open load
--(\)--	Close load
--[ ]--	Normally open switch
--[\]--	Normally close switch

2. **Control system flowchart (Function block diagram):**

Use the logic gates program PLC to control and simulate the circuit, and the gates are:

1. AND gate: must have two input at least, and this equivalent two switches on series.

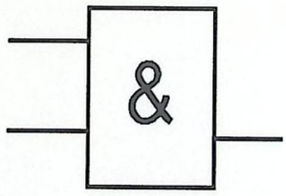


Figure 2.2: AND gate

2. OR gate: must have two input at least, and this equivalent two switches on parallel.

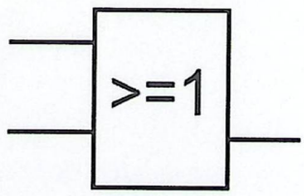


Figure 2.3: OR gate

3. NOT gate: has one input, and the output is inverting the input, implement a closed switch N.C.

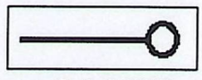


Figure 2.4: NOT gate

3. **Statement List (STL):**

Describing the circuit with the commands, and this method like writing assembly language, commands are:

1. (AND) series, labeled (A).
2. (OR) parallel, labeled (O).
3. (NOT) closed switch (N).
4. Parentheses implements parallel group.

2.3 Packaging Machine of Sugar Bags: Design and Operating:

Figure 2.5: Packaging machine of sugar bags

One kilogram of sugar bags packaging machine, contains 3 phase motor with 120 watt to drive the conveyor (2 meter) which moves the filled bag to package.

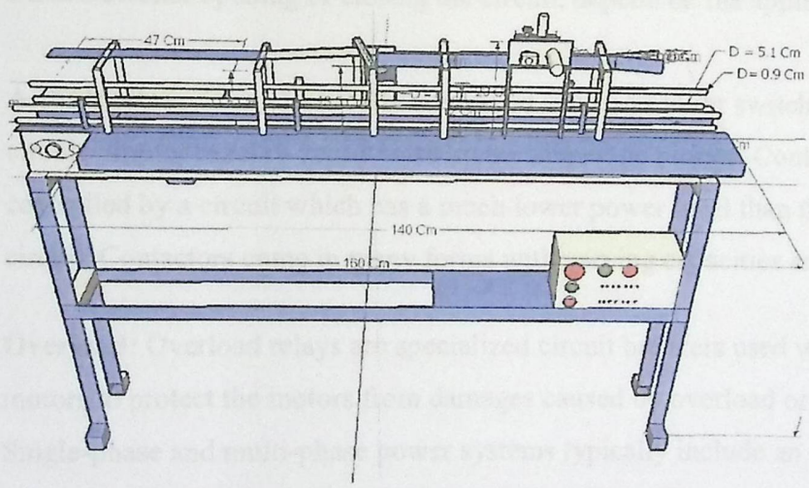


Figure 2.6: Packaging machine of sugar bags 3D

The bag on the conveyor reaches 2 double cylinders to make the first close and put the glue on. After that bag will reach to glue bottle that is beside the conveyor, here the glue will be put in the bag by some mechanism.

Finally the bag moves with some technique to be packaged and this will happen by using "PLC" controller.

- **Machines operating levels :**

1. Level one: Preparing the upper part of bag to move.
2. Level two: The bag reaches to second level of packaging to make the first close by using two double acting cylinders, when bag reaches the limit switch the motor will stop and the cylinders will activate by some arrangement.
3. Level three: Glue operation: here the bag reaches to the glue bottle, and it will be glued due to presence the tire on glue.
4. The last packaging operation: After ending the glue operation the bag will enter the last level, the bag forced to move under small piece of metal.

- **Machine devices:**

➢ **Input devices:**

1. **Push buttons:** opening or closing the circuit, depend on the application.

2. **A contractor:** is an electrically controlled switch used for switching a power circuit, similar to relay except with higher amperage ratings. Contactor is controlled by a circuit which has a much lower power level than the switched circuit. Contactors come in many forms with varying capacities and features.

3. **Overload:** Overload relays are specialized circuit breakers used with industrial motors to protect the motors from damages caused by overload or electrical faults. Single-phase and multi-phase power systems typically include an overload relay for interrupting power in the power conductors when a fault condition occurs, such as a ground fault, phase loss, overcurrent, or undercurrent condition.

4. **Limit switch:** Switches are commonly employed as input devices to indicate the presence or absence of a particular condition in a system or process that is being monitored and/or controlled, in motorized electromechanical systems; limit

switches provide the function of making and breaking electrical contacts and consequently electrical circuits.

5. **Emergency switch:** In factories and the like where industrial machinery is installed, in order to ensure the safety of an operator in cases such as where a fault occurs during operation of machinery, an emergency stop switch for emergency stop of the machinery is necessarily provided. A machine is typically powered by an electrical power source and typically has an on/off switch for use during normal operating conditions, for safety reasons, a machine will usually also include an emergency stop switch for terminating electrical power to the machine in an emergency situation.

➢ **Output devices:**

1. **Motor:** An **electric motor** is a type of machine that converts electrical energy into mechanical energy. Electric motors operate through interacting magnetic fields and current-carrying conductors to generate force, although a few use electrostatic forces. The reverse process, producing electrical energy from mechanical energy, is accomplished by an alternator, generator or dynamo. Many types of electric motors can be run as generators, and vice versa.

2. **Solenoids:** A solenoid is a coil wound into a tightly packed helix. In physics, the term solenoid refers to a long, thin loop of wire, often wrapped around a metallic core, which produces a magnetic field when an electric current is passed through it. Solenoids are important because they can create controlled magnetic fields and can be used as electromagnets, the term solenoid refers specifically to a magnet designed to produce a uniform magnetic field in a volume of space (where some experiment might be carried out).

3. **Relay indicators:** A relay is an electrically operated switch. Many relays use an electromagnet to operate a switching mechanism mechanically, but other

operating principles are also used. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.

4. Buzzer, pilot lamp and alarms devices used in this application.

2.4 PIC vs. PLC:

Table 2-2: PLC Vs PIC

No	characteristics	PLC	PIC
1	Power	80-240VAC	5VDC
2	I/O	Accept	Accept
3	Output Voltage	24V	5V
4	Price	Around (100 – 3000) $	Around 20 $
5	Versatile	Hared	Easer then PLC

A PLC is an industrial grade computer. A PLC can be supplied with power (usually 80-240VAC) and it will accept inputs, NPN or PNP, and it has outputs, relay, NPN, or PNP. There are of course other options. Some PLCs provide 24Vdc out for sensors. PLCs are usually very hard to design. The inputs and outputs are often protected and the unit is not very susceptible to electrical noise or interference. PLCs come in a variety of sizes and features and range greatly in price. A simple PLC might cost $100 and a large unit might cost over $3000. As a result, PLC is a complete system, probably using many chips, so it's larger, more expensive, and not as versatile.

A PIC on the other hand is a microcontroller. A PIC would need a suitable circuit built around it to work like a PLC. Because it runs off of a low DC voltage, usually around 5VDC and it has inputs and outputs which are usually TTL (some pins are open collector). You can't connect most sensors directly to a PIC; it usually can't drive a load over 20mA per pin. As a

result, PIC is a single chip computer, as such it's very small, very cheap (Almost cost $20), and very versatile.

From above we can choose a PIC to take the place of a PLC, with using helper circuits, from low coast and versatility of PIC it could be possible.

2.5 PIC18F4550:

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1640 originally developed by General Instrument's Microelectronics Division. The name PIC initially referred to "Peripheral Interface Controller".

PICs are popular with developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability that's why we use PIC family and we choose PIC18f4550 category for the characteristics described later but the main reason is that it support USB interface.

The microcontroller used is PIC18f4550 with:

- ✓ 40 pin package.
- ✓ Two External Clock modes, up to 40 MHz.
- ✓ 32k byte FLASH program memory.
- ✓ 2k byte data memory or registers ("Register File").
- ✓ 256 byte EEPROM (non volatile) data registers.
- ✓ Three external interrupts.
- ✓ USB V2.0 Compliant SIE.
- ✓ 1-Kbyte dual access RAM for USB.

2.5.1 Pin description:

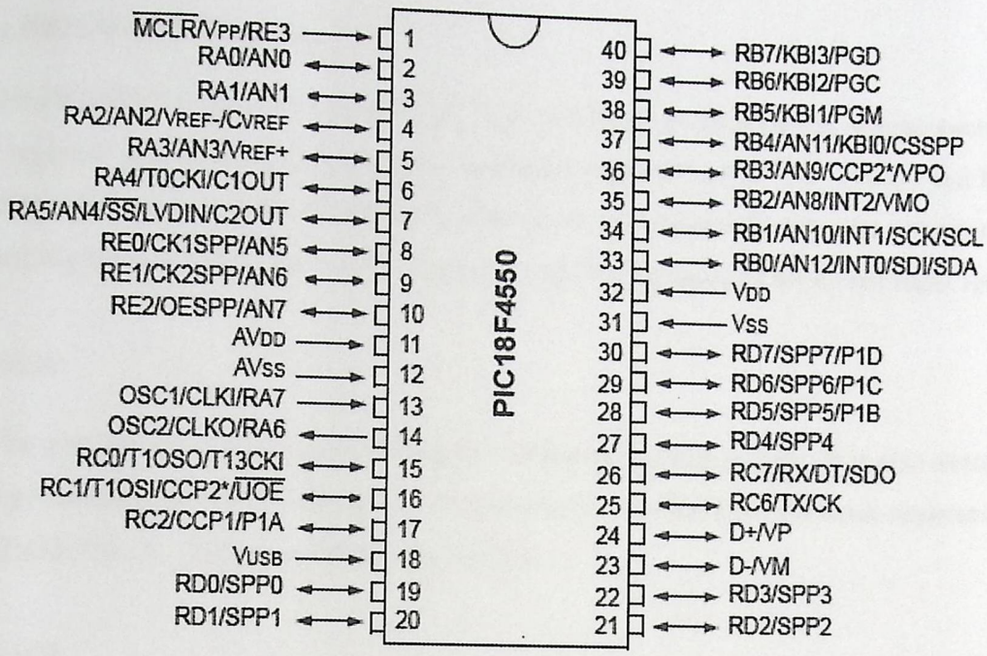


Figure 2.7: PIC18F4550

2.5.2 Clock oscillator and instruction cycle:

Any microcontroller is a complex electronic circuit, made up of sequential and combinational logic. At fantastic speed it steps in turn through a series of complex states, each state being dependent on the instruction sequence it is executing. While the detail of this process is invisible to us, it is still necessary to provide the 'clock' signal, a continuously running fixed frequency logic square wave. The overall speed of the microcontroller operation is entirely dependent on this clock frequency. It is not just the CPU that is dependent on the clock. In most microcontrollers many essential timing functions are also derived from it, ranging from counter/timer functions to serial communications. Furthermore, the overall power consumption of the microcontroller has a strong dependence on clock frequency; with high speed operation being much more power hungry than slow speed.

2.5.3 The PIC18F4550 I/O ports:

The PIC18F4550 have five ports A, B, C, D and E and it described as follow, each PORT has three registers associated for its operation and these registers is PORTX, TRISX and LATX, the port data itself appears in the PORTX, the data direction is determined by the bit values set in the (control registers) TRISX, the LATX is used to read the output data back (not input read).

- Port A:

The port can be used for general-purpose bi-directional digital data. It is also shared with the Analog functions, the Timer 0 inputs is shared with bit 4 of the Port, the three registers associated with Port A –PORTA, TRISA and LATA.

- Port B:

Port B is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. three primary registers – PORTB, TRISB and LATB, that bits 5–7 share with the in-circuit debug functions of PGD, PGC and PGM, the external interrupt can be made through PORTB pins.

- Port C:

Port C is a 7-bit wide, bidirectional port. The corresponding Data Direction register is TRISC. Its three primary registers – PORTC, TRISC and LATC, this port shares its pins with the serial ports and the CCP functions, while bit 0 is shared with the Timer 1 input. Port C is primarily multiplexed with serial communication modules.

- Port D:

Port D is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISD; the parallel port itself is Port D, with 3 bits of Port E being available for handshaking.

All pins on PORTD are implemented with Schmitt Trigger input buffers; each pin is individually configurable as an input or output.

- Port E:

Port E is a 4-bit wide port. The corresponding Data Direction register is TRISE.

2.5.4 PIC18F4550 timers:

Every timer needs a clock pulse to tick. The clock source can be internal or external. Internal clock, the 1/4th of the frequency of the crystal oscillator on the OSC1 and OSC2 pins (FOSC/4) is fed into the timer. Therefore, it is used for time delay generation, this is called a timer. External clock, fed pulses through one of the PIC's pins, this is called a counter.

Many of the PIC18 timers are 16 bits wide, each 16-bit timer is accessed as two separate register, low byte (TMRxL) and high byte (TMRxH). Each timer also has the TCON (Timer Control) register for setting modes of operation.

The 18F4550 have five programmable timers (timer0, timer1, timer2, timer3), as well as a Watchdog Timer, the general used counting and timing.

2.5.5 Analog-to-Digital Convertor (ADC):

The Analog-to-Digital (A/D) converter module has 10 inputs for the 28-pin devices and 13 for the 40/44-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

- The module has five registers:

  • A/D Result High Register (ADRESH)
  • A/D Result Low Register (ADRESL)
  • A/D Control Register 0 (ADCON0)
  • A/D Control Register 1 (ADCON1)
  • A/D Control Register 2 (ADCON2)

The ADCON1, TRISA, TRISB and TRISE registers all configure the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level will be converted.

The following figure shows all peripherals that are explained in this chapter:

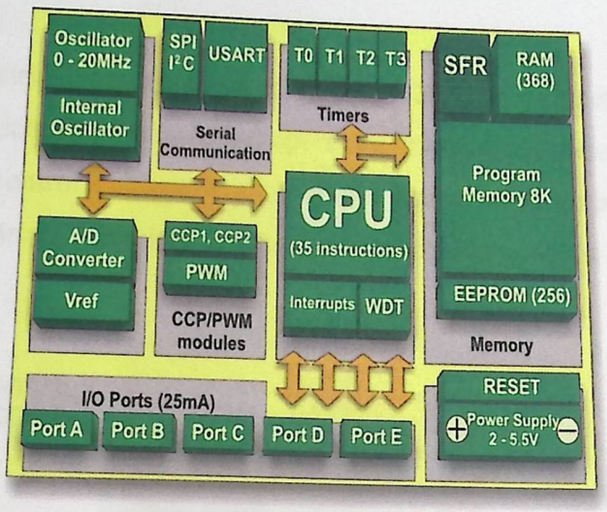


Figure 2.8: PICf184550
internal circuits

2.5.6 Programming language:

PIC18 microcontroller can be programmed in Assembly, C or a combination of the two. Other high level programming languages can be used but embedded systems software is primarily written in C.

The best suggestion is to write the code completely in C because it is much easier than writing your code in Assembly or a combination of languages. Despite the speed of the Assembly language than C, but C in more readability, so much easier to debug. In some case the using of combination of C and Assembly is mandatory when programming the Timer and interrupt.

2.6 Relay Connections:

A **relay** is an electrically operated switch. Many relays use an electromagnet to operate a switching mechanism mechanically, but other operating principles are also used. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.

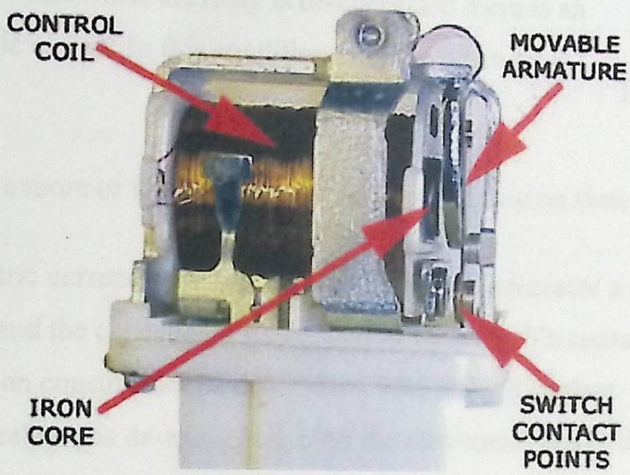


Figure 2.9: Relay

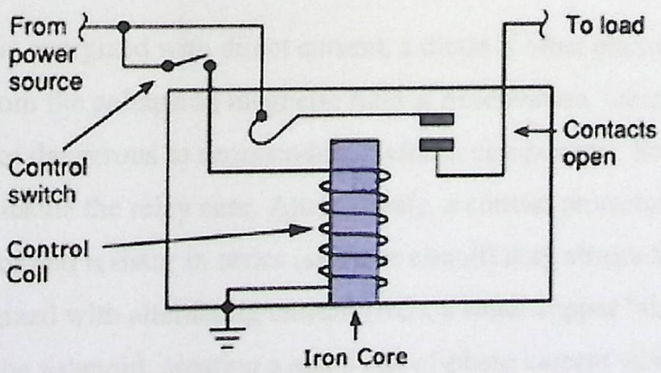


Figure 2.10: Relay circuit

2.6.1 Basic design and operation:

A simple electromagnetic relay consists of a coil of wire surrounding a soft iron core, an iron yoke which provides a low reluctance path for magnetic flux, a movable iron armature, and one or more sets of contacts . The armature is hinged to the yoke and mechanically linked to one or more sets of moving contacts. It is held in place by a spring so that when the relay is de-energized there is an air gap in the magnetic circuit. In this condition, and the other set is open.

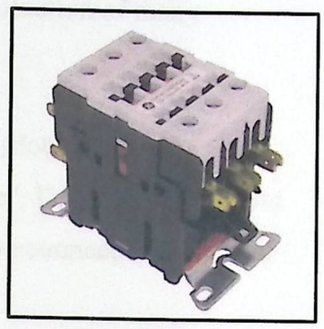


Figure 2.11: Relay contactor

Other relays may have more or fewer sets of contacts depending on their function.

When an electric current is passed through the coil it generates a magnetic field that attracts the armature and the consequent movement of the movable contact either makes or breaks (depending upon construction) a connection with a fixed contact. If the set of contacts was closed when the relay was de-energized, then the movement opens the contacts and breaks the connection, and vice versa if the contacts were open. When the current to the coil is switched off, the armature is returned by a force, approximately half as strong as the magnetic force, to its relaxed position. Usually this force is provided by a spring, but gravity is also used commonly in industrial motor starters. Most relays are manufactured to operate quickly. In a low-voltage application this reduces noise; in a high voltage or current application it reduces arcing.

When the coil is energized with direct current, a diode is often placed across the coil to dissipate the energy from the collapsing magnetic field at deactivation, which would otherwise generate a voltage spike dangerous to semiconductor circuit components. Some automotive relays include a diode inside the relay case. Alternatively, a contact protection network consisting of a capacitor and resistor in series (snubber circuit) may absorb the surge. If the coil is designed to be energized with alternating current (AC), a small copper "shading ring" can be crimped to the end of the solenoid, creating a small out-of-phase current which increases the minimum pull on the armature during the AC cycle.

2.6.4 Relay application considerations:

Selection of an appropriate relay for a particular application requires evaluation of many different factors:

- Number and type of contacts – normally open, normally closed, (double-throw)
- Contact sequence – "Make before Break" or "Break before Make". For example, the old style telephone exchanges required Make-before-break so that the connection didn't get dropped while dialing the number.
- Rating of contacts – small relays switch a few amperes, large contactors are rated for up to 3000 amperes, alternating or direct current.
- Voltage rating of contacts – typical control relays rated 300 VAC or 600 VAC, automotive types to 50 VDC, special high-voltage relays to about 15 000 V
- Coil voltage – machine-tool relays usually 24 VAC, 120 or 250 VAC, relays for switchgear may have 125 V or 250 VDC coils, "sensitive" relays operate on a few mill amperes.
- Coil current.
- Package/enclosure – open, touch-safe, double-voltage for isolation between circuits, explosion proof, outdoor, oil and splash resistant, washable for printed circuit board assembly.
- Mounting – sockets, plug board, rail mount, panel mount, through-panel mount, enclosure for mounting on walls or equipment.
- Switching time – where high speed is required.
- Coil protection – suppress the surge voltage produced when switching the coil current
- Isolation between coil circuit and contacts.
- Aerospace or radiation-resistant testing, special quality assurance.

2.7 Limit Switch:

Limit Switches a mechanical limit switch interlocks a mechanical motion or position with an electrical circuit. A good starting point for limit-switch selection is contact arrangement. The most common limit switch is the single-pole contact block with one NO and one NC set of contacts; however, limit switches are available with up to four poles.

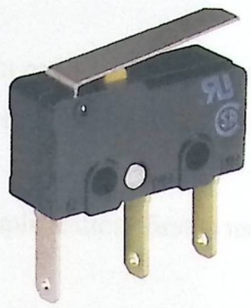


Figure 2.12: Limit switch

Limit switches also are available with time-delayed contact transfer. This type is useful in detecting jams that cause the limit switch to remain actuated beyond a predetermined time interval.

Other limit switch contact arrangements include neutral-position and two-step. Limit switches feature a neutral-position or center-off type transfers one set of contacts with movement of the lever in one direction. Lever movement in the opposite direction transfers the other set of contacts. Limit switches with a two-step arrangement, a small movement of the lever transfers one set of contacts, and further lever movement in the same direction transfers the other set of contacts.

Maintained-contact limit switches require a second definite reset motion. These limit switches are primarily used with reciprocating actuators, or where position memory or manual reset is required. Spring-return limit switches automatically reset when actuating force is removed.

2.8 Amplification Circuits:

Using amplification circuit is an important step in our project, in order to the voltage which PIC microcontroller outputting is less than the voltage and current needed to operate the relays and contactors.

We here have two options to thinking in current and voltage amplification, first is user an operational amplifiers and that is cost us amplifier for each output pin.

The second choice is using ULN2804 integrated circuit as a solution for operating high voltage of relays and contactors, and I will talk briefly about this IC.

2.8.1 ULN2804:

Featuring continuous load current ratings to 500 mA for each of the drivers, the Series ULN28xxA/LW and ULQ28xxA/LW high voltage, high-current Darlington arrays are ideally suited for interfacing between low-level logic circuitry and multiple peripheral power loads. Typical power loads totaling over 260 W (350 mA x 8, 95 V) can be controlled at an appropriate duty cycle depending on ambient temperature and number of drivers turned on simultaneously. Typical loads include relays, solenoids, stepping motors, magnetic print hammers, multiplexed LED and incandescent displays, and heaters. All devices feature open-collector outputs with integral clamp diodes.

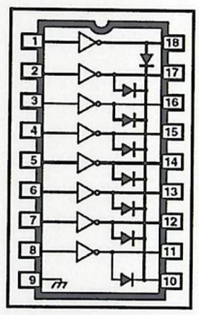


Figure 2.13: ULN2804

The ULN2804 have series input resistors selected for operation directly with 5 V TTL or CMOS. These devices will handle numerous interface needs particularly those beyond the capabilities of standard logic buffers.

The ULN2804, have series input resistors for operation directly from 6 V to 15 V CMOS or PMOS logic outputs.

The ULN2804 are the standard Darlington arrays. The outputs are capable of sinking 500 mA and will withstand at least 50 V in the off state.

Outputs may be paralleled for higher load current capability. The ULN2804 will withstand 95 V in the off state.

These Darlington arrays are furnished in 18-pin dual in-line plastic packages (suffix 'A') or 18-lead small-outline plastic packages (suffix 'LW'). All devices are pinned with outputs opposite inputs to facilitate ease of circuit board layout. Prefix 'ULN' devices are rated for operation over the temperature range of -20°C to +85°C; prefix 'ULQ' devices are rated for operation to -40°C.

- **Why we chose ULN2804:**

1. TTL, DTL, PMOS, or CMOS Compatible Inputs.
2. Output Current to 500 mA.
3. Output Voltage to 95 V.
4. Transient-Protected Outputs.
5. Dual In-Line Package or Wide-Body Small-Outline Package.
6. Easy to use and interface with PIC18f4550.

CHAPTER THREE

SYSTEM CONCEPTUAL DESIGN

3.1 Overview

3.2 Detailed System Objectives

3.3 General block diagram

3.4 System block parts

3.5 Software blocks option

3.1 Overview:

In this chapter we talk about our project diagrams and our choices to designing the system.

3.2 Detailed System Objectives:

1. Execute all controlling statements used by PLC which replaced with PIC.
2. Interfacing PIC microcontroller with the machine controlling system.
3. Translating PLC statement list program into C program for PIC.
4. Convenient PIC microcontroller with the system without any changing in operating mechanism of packaging machine.
5. Avoiding high price of PLC.
6. Trying to apply connection between the machine and outside world.

There are options in a design of our project to archive the idea and produce a closed bag of sugar; we will build our choices and different module to reach for a complete solution.

3.3 General block diagram:

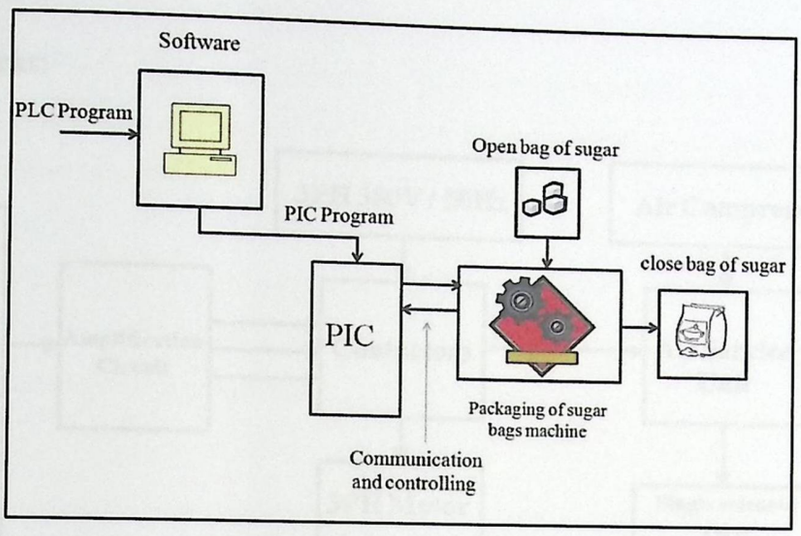


Figure 3.1: General Block Diagram

The project can be divided into three main blocks. First, the software block translates the PLC program into a PIC program in order to apply it.

The second block is a PIC block that controls in the system in order to ensure accuracy and responsiveness that we need, without changing the major idea of machine's project.

The third block is one of the projects which are done in the Palestine Polytechnic University (PPU) using a PLC system to control the mechanical parts of the machine. The project is titled by "packaging machine of sugar bags design and operation".

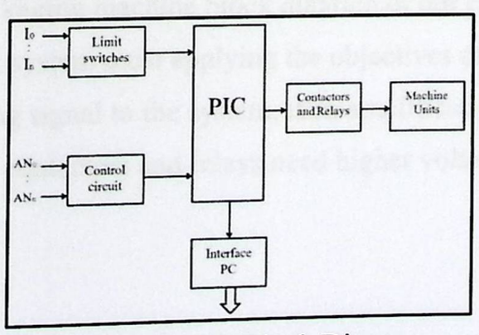


Figure 3.2: PIC Block Diagram

The figure 3.2 PIC block diagram shows limit switches I0 - In as input modules and signals channels AN0 – ANn, and output modules as relays and contactor which controlling with the system, interfacing with the computer by programmer and USB.

3.4 system blocks:

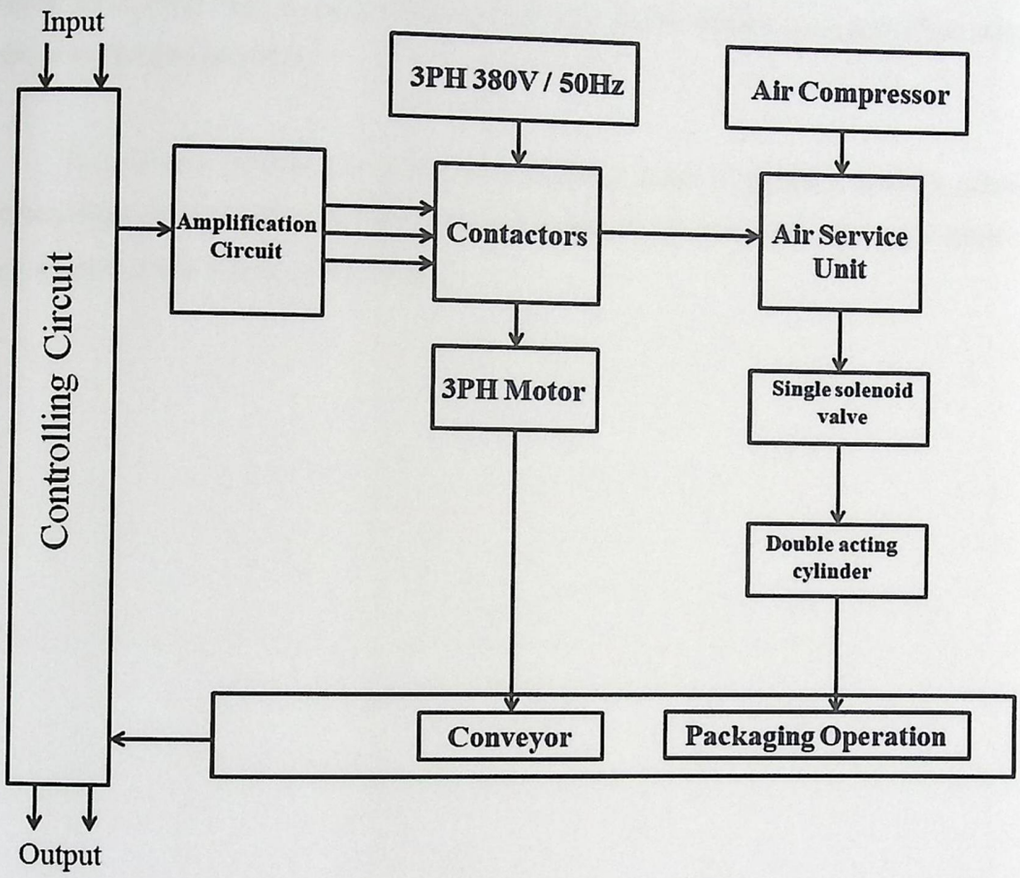


Figure 3.3: Machine Block Diagram

Figure 3.3 shows the packaging machine block diagram of our project, which explain system behavior, as whole block system must applying the objectives of packaging machine, PIC microcontroller output controlling signal to the system, then amplification circuits will amplifying the voltage since all contactors and relays need higher voltage than the microcontroller outputted.

The mission of contactors is connect the high voltage to 3PH motor which given from out electricity source, due to controlling with the conveyor movement and to stop its when the system need that and operate it again to carrying sugar bag between different levels.

PIC controlling with air compressor, and air unit is an important level from other levels to reach in the end closed bag of sugar, vulvas of air must know when it work and when stop, all that is done by microcontroller.

Include two double acting cylinders (cylinder control cylinder), used for achieve the initial discourage of bags and to get discourage the bags as desired form, and to get some feedback signals when it begin and finish.

3.5 Software blocks option:

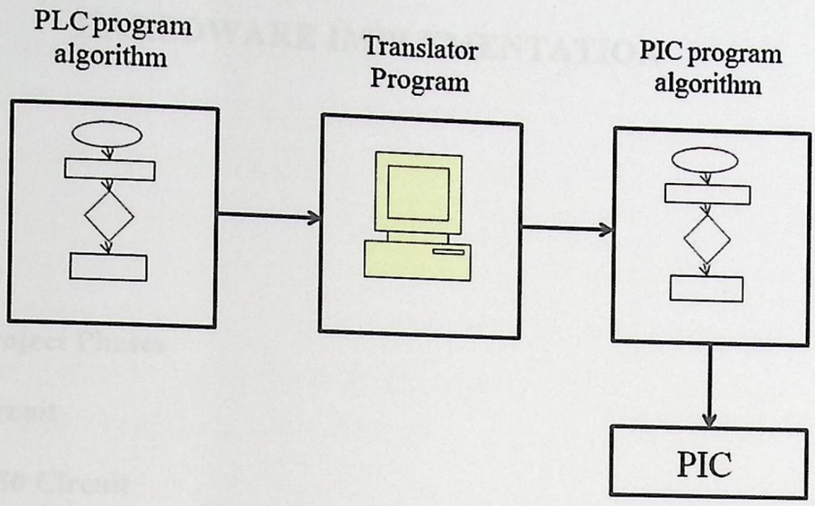


Figure 3.4: Software blocks

Figure 3.4 shows the software block what we needed to translate from PLC language to PIC language, and we chose statements list for PLC translating to C language, after that C file need an IDE like MPLAB to editing and compiling, finally export a hex file downloaded to PIC .

We chose C#.net as a visaul prgraming language using .net technology to produce our translator program, and our choicing C#.net refer to :

1. C# is strong language.
2. C# with .net can esaily design visaul inter face.
3. C# is a common syntax and implimentation.
4. Has a huge library and multiple option facelation programming.

CHAPTER FOUR

HAREDWARE IMPLEMENTATION

4.1 Project Phases

4.2 Description Project Phases

4.2.1 Power Circuit

4.2.2 PIC18f4550 Circuit

4.2.3 ULN 2003 Circuit

4.2.4 Relay Circuit

4.2.5 Limit Switch Circuit

4.3 Computer Connection (PICkit 2)

4.1 Project Phases:

The project will be design in phases as follows:

- Design and construction the system circuit.
  - PIC circuits (Power, Grounded, and Reset).
  - Relays circuits and interfacing PIC.
  - Limit switch circuit.

- Application program.
  - The main application that make languages translation.
  - Program will operate the microcontroller.

4.2 Description project phases:

Phases will be designed according to the block diagram Figure (3.3) described in chapter three.

4.2.1 Power circuit:

We have two power resources one for the ICs and other for the machine loads and we will describe later.

As we know ICs need +5 DC volts as source power and to supply our circuit with that we use converter 220 volts AC to 24 volts AC or DC.

For AC there is rectification circuit and regulation to get +5 volts and the following figure explain that.

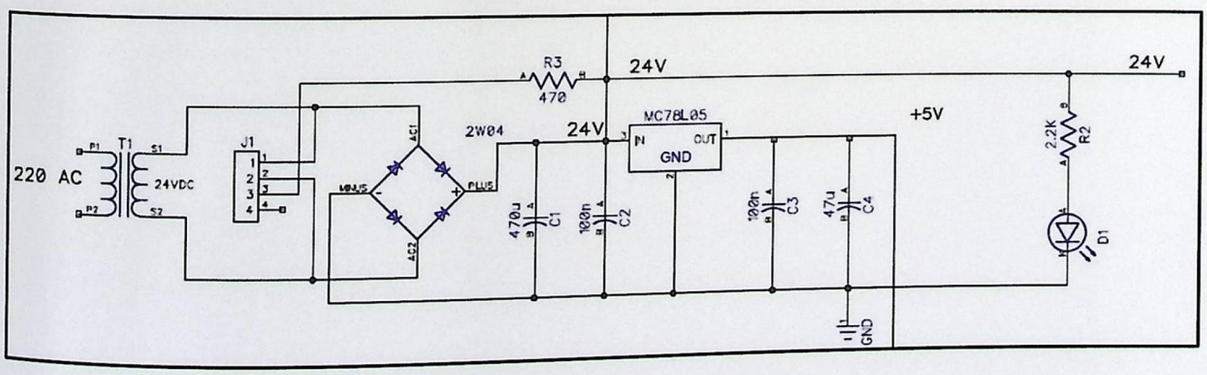


Figure 4.1: Power circuit

Using transformer from 220 AC volts to 24 DC volts to feeding all circuit elements like ICs and LEDs and relays.

Connecting 24 Volts power to junction J1 (1 and 2) and the third terminal is just for diode emitting, power may be AC or DC, if the power AC then bridge BR1 responsible about rectification the current to DC.

The capacitor C1 will remove the ripples to convert the current to direct, this voltage will use it for 24DC volts relays, and before arrives the regulator.

The input capacitor C2 and three terminal regulator and output capacitor C3 connected to reduce voltage to +5 volte for feeding PIC and ULN.

The input capacitor C4 filtering the current from any noise, resistance R2 and LED D1 just to showing the circuit is active.

4.2.2 PIC18f4550 circuit:

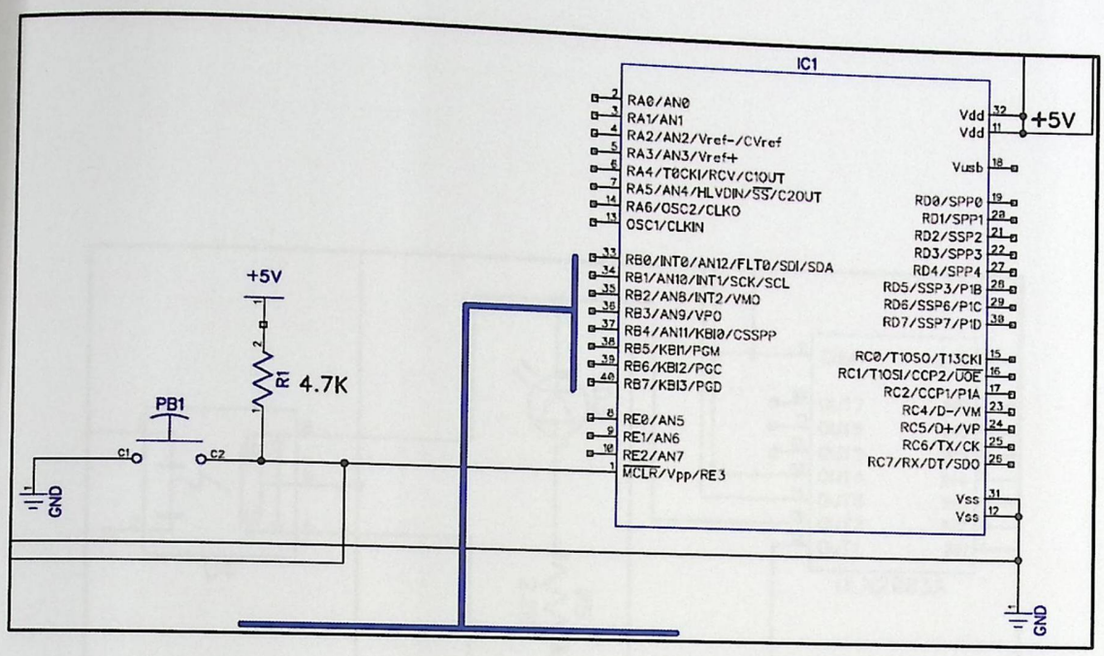


Figure 4.2: PIC circuit

PIC microcontroller operating on +5 volts so we connect Vdd pins 32 and 11 to +5 volts for feeding, and the pins Vss 31 and 12 connected to ground.

Reset circuit from the first pin MCLR low voltage active for that we connect pull up resistance to +5 volts and dip switch to ground, when circuit open MCLR = 1, when close dip switch MCLR = 0 and will activate PIC reset.

PORTB will use it as output port and connect to relays through ULN IC since relays is 24 DC volts.

PORTA will use it as input port for limit switches to get circuit open or close this value equal zero or one and we will explain limit switches circuit later.

4.2.3 ULN 2003 circuit:

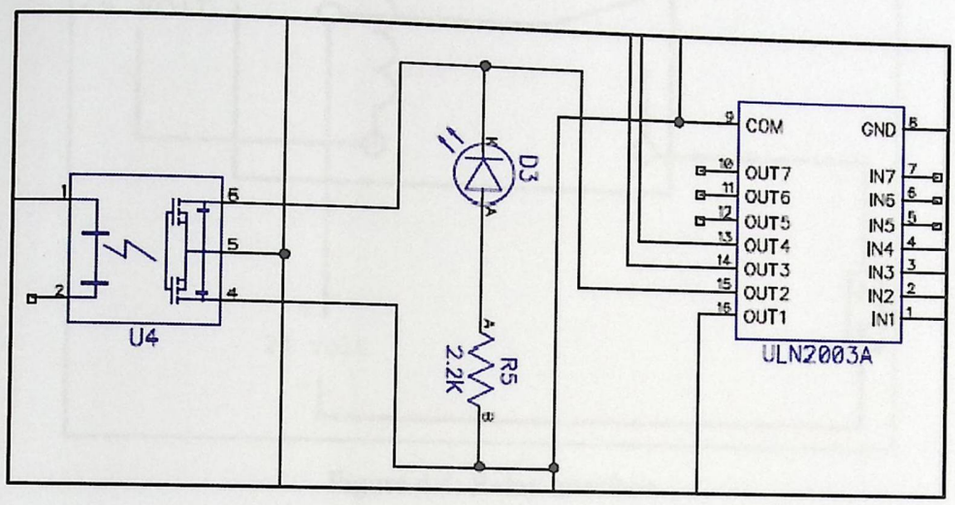


Figure 4.3: Relay circuit

ULN2003 is high voltage, high current darlington arrays containing seven open collector darlington pairs with common emitters. Each channel rated at 500mAand can withstand peak currents of 600mA.

Gets input +5 volts form PIC and output go to relay with 24 volts, using resistance and LED for each relay just for indication.

4.2.4 Relay circuit:

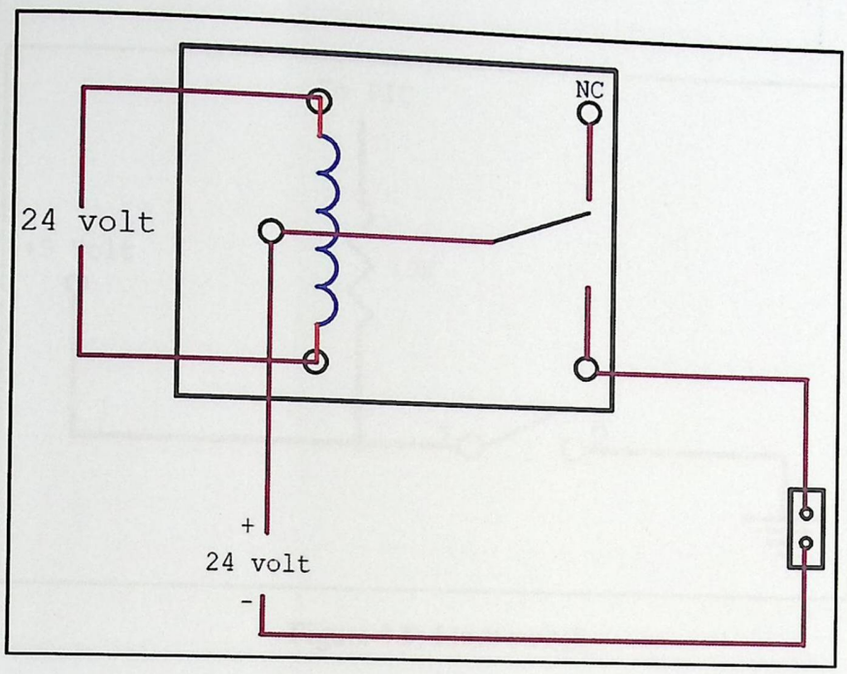


Figure 4.4: Relay interface

With normally open relay the circuit is open and on power for the load, when PIC issue orders to relay and voltage of 24 volt came to coil the circuit will close.

The load feed with 24 volt, for indication here this source of voltage is different from the power of PIC duo to loads will not able to operate on the same power resource sharing with PIC.

All relays connected to PORTB as output port, we have four relay in order to the machine has three solenoid and one for the contractor who is controlling with 3 phase motor.

- First relay => motor.
- Second relay => First solenoid.
- Third relay => Second solenoid.
- Forth relay => Third solenoid.

4.2.5 Limit switch circuit:

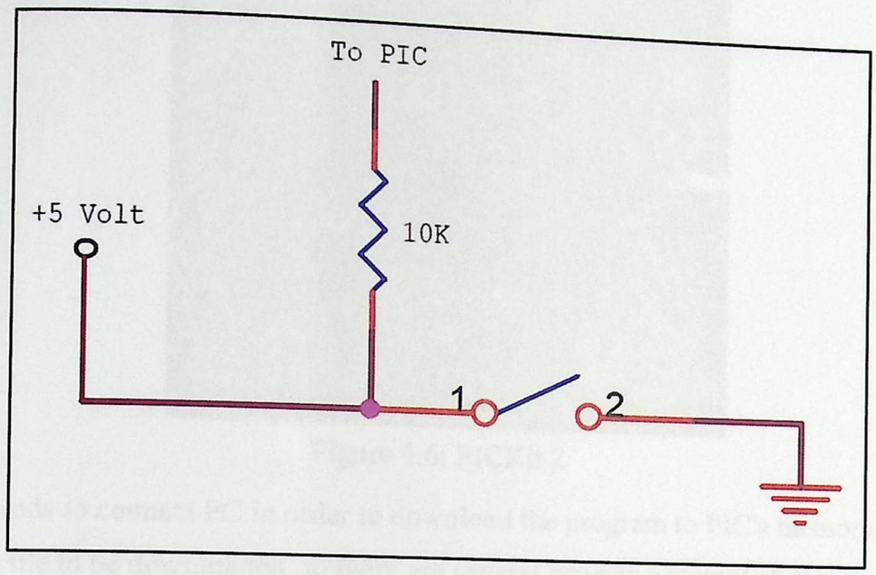


Figure 4.5: Limit switch

We have three limit switches in the machine so all input values connected to PORTA, this circuit enable us take value one or zero.

When limit switch is open as in figure 4.5 PIC will reads value is one, otherwise when switch closed PIC reads value zero duo to the 10 K resistance.

Switches are important elements in this project in order to controlling in motor and the vertical cylinder and the horizontal cylinder.

The scene like this in the beginning first relay is closed who is controlling motor, motor always operate unless the first limit switch is closed and that is when sugar bags arrive just under the cylinder, after that horizontal cylinder will be pushed up to second limit switch closed, then vertical cylinder will be pushed vertically up to third limit switch closed.

4.3 Computer Connection (PICkit2):

Figure 4.6: PICKit 2

PIC needs to connect PC in order to download the program to PIC's memory; here we mean the hex file to be downloaded, so there are several ways to accomplish this work.

In our case we used PICkit 2 programmer to connect PC and programming PIC, from PC a USB connection used and connect 5 bins in PIC (1 VPP, 40 PGD, 39 PGC, 11 Vdd, and 12 GND), and compatible with MPLab platform.

The PICkit 2 Development Programmer/Debugger is a low-cost development programmer. It is capable of programming most of Microchip.s Flash microcontrollers and serial EEPROM devices.

The USB port connection is a USB mini-B connector. Connect the PICkit 2 to the PC using the supplied USB cable.Calibration allows greater accuracy both in the VDD voltage supplied to the target from PICkit 2, and the voltage detected on a powered target and reported in the software. The calibration is stored in the PICkit 2 unit nonvolatile memory, so the unit will remain calibrated even when used from within MPLAB IDE.

Any other details will mention it in Appendix.

CHAPTER FIVE
SOFTWARE IMPLEMENTATION

5.1 Overview

5.2 The Software's

5.2.1 C#.net programming language

5.2.2 C programming language

5.1 Overview:

This program is an integrating part II of the project, and it is the software part, and this program is doing translation from the language of the PLC to C.

The following flow chart illustrates the general idea of software:

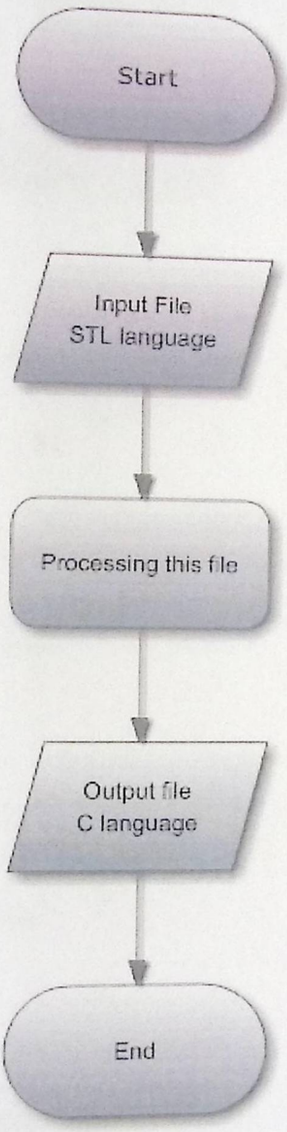


Figure 5.1: General Flowchart

In the treatment process there are several steps in the software translating and this flow chart demonstrate some of these steps:

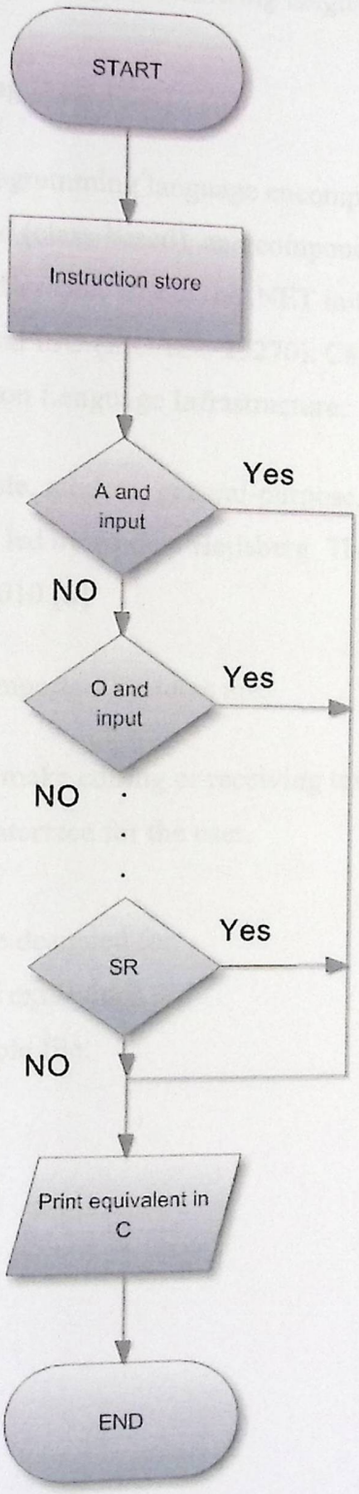


Figure 5.2: Software Flowchart

5.2 The Software's:

The software will be written using the following languages:

5.2.1 C#.net programming language:

C# is a multi-paradigm programming language encompassing imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure.

C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 4.0, which was released on April 12, 2010.[8]

This program has several important features are:

1. Has a text editor sheet, make editing or receiving text from saved file.
2. And he has a familiar interface for the user.
3. Easy to use.
4. Doing the work what he designed for.
5. Export files in text or C extensions.
6. The output file is readable file.

7. Demo version.

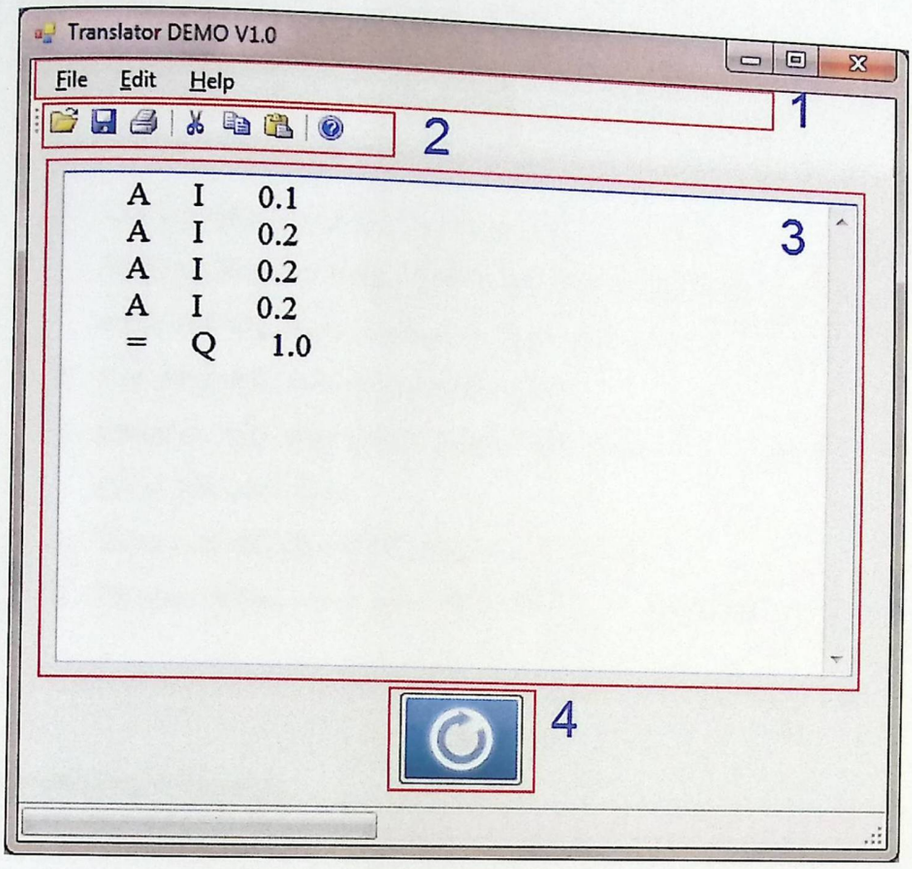


Figure 5.3: Translator Interface

1. Menu bar has file and edit and help.

   - File: Open File, Save File, Exit.
   - Edit: editing text such copy and paste.
   - Help: information about program.

2. Shortcuts icons.
3. Writing and editing text area.
4. Convert button.

- How to use it:

The program has easy to use, and is a familiar interface and to use it must follow the instructions:

1. Use the editor to write statements list language of PLC or import a text file, language must be compatible with Siemens Simatic Step7 program and it static writing, mean the spaces between instruction must be consider.
2. Converting button makes translating.
3. Program asks you to save output file, choose a directory and file's extension (Text file or C file).
4. Then everything ends if there are no errors.
5. Program make some error detection like if there is no text in text area.

5.2.2 C programming language:

PIC can be programmed using different languages including Basic language, Assembly language, and C programming language. We chose to program the PIC microcontroller using C language since they are familiar with it, and it can deal well with the complex computation the project needs.

CHAPTER SIX

TESTING

6.1 Software Testing

6.2 Hardware Testing

6.1 Software Testing:

We tested the software with some programs of STL and the following is input files and output files using this software:

1. Four input series switches and one output load:

- Input file:

```
A   I   0.1
A   I   0.2
A   I   0.3
A   I   0.4
=   Q   1.0
```

- Output file:

```
#include<p18f4550.h>
#pragma config WDT=OFF
#pragma config FOSC=INTOSC_HS
#pragma config LVP = OFF
void main(void)
{
int S = 0;
int R = 0;
OSCCON=OSCCON&0b11111111;
ADCON1=15;
TRISA = 255; // input port
TRISB = 0;  // output port
while(1)
```

```
{

if(PORTAbits.RA2 == 0  && PORTAbits.RA3 == 0  && PORTAbits.RA0 == 0  &&
PORTAbits.RA1 == 0 )

{

PORTBbits.RB0=1;

}

}

}
```

2. Four input parallel switches and one output load:

- Input file:

```
O   I   0.1

O   I   0.2

O   I   0.3

O   I   0.4

=   Q   1.0
```

- Output file:

```
#include<p18f4550.h>

#pragma config WDT=OFF

#pragma config FOSC=INTOSC_HS

#pragma config LVP = OFF

void main(void)

{

int S = 0;
```

```
int R = 0;

OSCCON=OSCCON&0b11111111;

ADCON1=15;

TRISA = 255; // input port

TRISB = 0;  // output port

while(1)

{

if(PORTAbits.RA2 == 0  || PORTAbits.RA3 == 0  || PORTAbits.RA0 == 0  || PORTAbits.RA1
== 0 )

{

PORTBbits.RB0=1;

}

}

}
```

3. The machine file that operate PLC:

- Input File:

```
Network 1
    A   I    0.1
    =   Q    1.1
Network 2
    A   I    0.6
    S   Q    1.1
    A   I    0.4
    R   Q    1.1
    A   Q    1.1
    =   Q    1.1
Network 3
    A   I    0.5
    S   Q    1.3
    A   I    0.7
    R   Q    1.3
    A   Q    1.3
    =   Q    1.3
Network 4
    A   I    0.5
    S   Q    1.2
    A   I    0.6
    R   Q    1.2
```

```
  A   Q   1.2
  =   Q   1.2
Network 5
  A   I   0.4
  S   Q   1.1
  A   I   0.7
  R   Q   1.1
  A   Q   1.1
  =   Q   1.1
```

- Output file:

```
1 #include<p18f4550.h>
2 #include<delays.h>
3 #pragma config WDT=OFF
4 #pragma config FOSC=INTOSC_HS
5 #pragma config LVP = OFF
6 void main(void)
7 {
8 int S = 0;
9 int R = 0;
10 OSCCON=OSCCON&0b11111111;
11 ADCON1=15;
12 TRISA = 255; // input port
13 TRISB = 0;  // output port
```

```
14 while(1)
15{
16
17 if(PORTAbits.RA0==1)
18{
19 PORTBbits.RB0=1;
20 }
21
22 ////////////////1/////////////////
23
24 S = 0;
25 R = 0;
26 if(PORTAbits.RA2 == 0 )
```

```
27 {
28 S = 1 ;
29 }
30 if(PORTAbits.RA0 == 0 )
31 {
32 R = 1 ;
33 }
34 if( S == 0 && R == 1 )
35 {
36 PORTBbits.RB0 = 0 ;
37 }
38 if( S == 1 && R == 0 )
39 {
40 PORTBbits.RB0 = 1 ;
41 }
42 if( S == 1 && R == 1 )
43 {
44 PORTBbits.RB0 = 0 ;
45 }
46
47 ////////////2////////////
48
49 S = 0;
50 R = 0;
```

```
51 if(PORTAbits.RA2 == 0 )
52 {
53 S = 1 ;
54 }
55 if(PORTAbits.RA5 == 0 )
56 {
57 R = 1 ;
58 }
59 if( S == 0 && R == 1 )
60 {
61 PORTBbits.RB3 = 0 ;
62 }
63 if( S == 1 && R == 0 )
64 {
65 PORTBbits.RB3 = 1 ;
66 }
67 if( S == 1 && R == 1 )
68{
69 PORTBbits.RB3 = 0 ;
70 }
71
72 ////////////3////////////
73
74 S = 0;
```

```
75 R = 0;
76 if(PORTAbits.RA1 == 0 )
77 {
78 S = 1 ;
79 }
80 if(PORTAbits.RA2 == 0 )
81 {
82 R = 1 ;
83 }
84 if( S == 0 && R == 1 )
85 {
86 PORTBbits.RB2 = 0 ;
87 }
88 if( S == 1 && R == 0 )
89 {
90 PORTBbits.RB2 = 1 ;
91 }
92 if( S == 1 && R == 1 )
93 {
94 PORTBbits.RB2 = 0 ;
95 Delay10TCYx(5);
96 PORTBbits.RB0 = 1 ; // run m
97 Delay10KTCYx(1);
98 }
```

```
99
100 ////////////4////////////
101
102 S = 0;
103 R = 0;
104 if(PORTAbits.RA0 == 0 )
105 {
106 S = 1 ;
107 }
108 if(PORTAbits.RA5 == 0 )
109 {
110 R = 1 ;
111 }
112 if( S == 0 && R == 1 )
113 {
114 PORTBbits.RB1 = 0 ;
115 }
116 if( S == 1 && R == 0 )
117 {
118 PORTBbits.RB1 = 1 ;
119 }
120 if( S == 1 && R == 1 )
121 {
122 PORTBbits.RB1 = 0 ;
```

```
123 }
124
125 ////////////5/////////////////
126
127
128 S = 0;
129 R = 0;
130 if(PORTAbits.RA0 == 0 )
131 {
132 S = 1 ;
133 }
134 if(PORTAbits.RA4 == 0 )
135 {
136 R = 1 ;
137 }
138 if( S == 0 && R == 1 )
139 {
140 PORTBbits.RB1 = 0 ;
141 }
142 if( S == 1 && R == 0 )
143 {
144 PORTBbits.RB1 = 1 ;
145 }
146 if( S == 1 && R == 1 )
```

```
147 {
148 PORTBbits.RB1 = 1 ;
149 }
150 }
151
```

1. Line 1 Include the library of PIC18F4550.
2. Line 2 Include Delay library.
3. Line 3 Watch dog timer is of since we using direct electricity source.
4. Line 4 Internal oscillating.
5. Line 5 Low voltage sensor is off.
6. Line 6 Main function.
7. Line 7 Start main function.
8. Line 8 Define S as variable, S is Set.
9. Line 9 Define R as variable, R is Reset.
10. Line 10 Determine the oscillating with 8 MHz.
11. Line 11 All inputs and outputs are digital.
12. Line 12 Direction of A PORT is input.
13. Line 13 Direction of B PORT is output.
14. Line 14 Start while, to run the program all the time.
15. Line 95 to 97 we add this line to complete the machine work.
16. All other lines are translating the switches and SR in STL.

6.2 Hardware Testing:

The hardware implemented and tested as in the previous chapter and the machine worked properly. To watch our video trailer about the machine you can visit youtube site in this link: http://www.youtube.com/watch?v=lOchc82og6g

CHAPTER SEVEN

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

7.2 Problems

7.2.1 Hardware problems

7.2.2 Software Problems

7.3 Recommendation

When starting the realization practically, the nature and the challenges of the project became more and more obvious. Many problems appeared and we managed to overcome them, however, many lessons were learned.

7.1 Conclusions:

1. In any project or work, the designer always tries to design his project with the lowest cost and the best performance or may be the same performance, as like the "Replacing PLC controller with PIC microcontroller" project, the sugar machine works in the same performance but the cost is less than using of PLC controller in the "packaging machine of sugar bags design and operation" project, with the same working condition in the industrial environment.
2. We design our controller to appropriate with the packaging machine, and that is a prove for any other automated projects to be done with this own controller.

7.2 Problems:

There are two kinds of problem that the "Replacing PLC controller with PIC microcontroller" project faced, that is hardware and software problems.

7.2.1 Hardware problems:

1. The difference of power supply between the PIC circuit and PLC and the PIC circuit itself, where PLC supplied 24 V, and the PIC circuit itself ranges between 5 or 24 V. The difference of power supply in the PIC circuit led to damage the PIC itself and ULN2003 microchips.
2. Power supply of programmer PICkit2, when you programming PIC using one power supply and that are from USB only; don't leave the power supply of PIC ON.
3. Some of the pipe air pressure of the sugar packaging machine project is not closed tightly, so the air may leak, and this may be effect on air compression and the movement of the arm, this forced us to reinstall the pressure pipes again.

7.2.2 Software Problems:

1. It was difficult that we do all STL instructions.
2. The great length of the program was difficult for us to detect programming errors.
3. Difficulty of finding equivalent sentences in C for each instruction in STL.
4. Problem with give numbers for ports, it was in a random way.
5. Dealing with interfaces in the visual studio was easy, but needs time.

7.3 Recommendations:

1. In this project, the performance that we choose is the same performance in the PLC circuit project, but any designer can increase the performance, so the production process may be faster than the last one.
2. The project may be controlled as remote control by using the internet, this is after adding a port that can connect the machine with the internet by using of a serial port, but this may be cost because of the using of a computer connected to PIC circuit by USB port.
3. In the software of the project is demo version, not all instruction of STL is entered in. But we use the instruction that cover packaging machine, you can see all instructions in appendix.
4. Packaging machine improvements; there is some improvements on the machine makes increasing the performance :
    - First we make timed the first switch; switch that stop the motor in order to stop the machine after some time without passing any bags of sugar.
    - May interfacing level sensor on glue box to tell us if the glue done.
    - May another sensor on the sugar bags container to tell us if the contender filled.

References

1. Hugh Jack – " Automating Manufacturing Systems with PLCs" - Version 4.7- April 14, 2005.

2. Martin P. Bates - "Programming 8-bit PIC microcontrollers in C".

3. Abd Al Rahman Zamar'a, Mohammad Garaja, mohammad Alwawi – "Packaging Machine of Sugar Bags: Design and Operation" – 2009.

4. http://www.circuitstoday.com/category/relay.

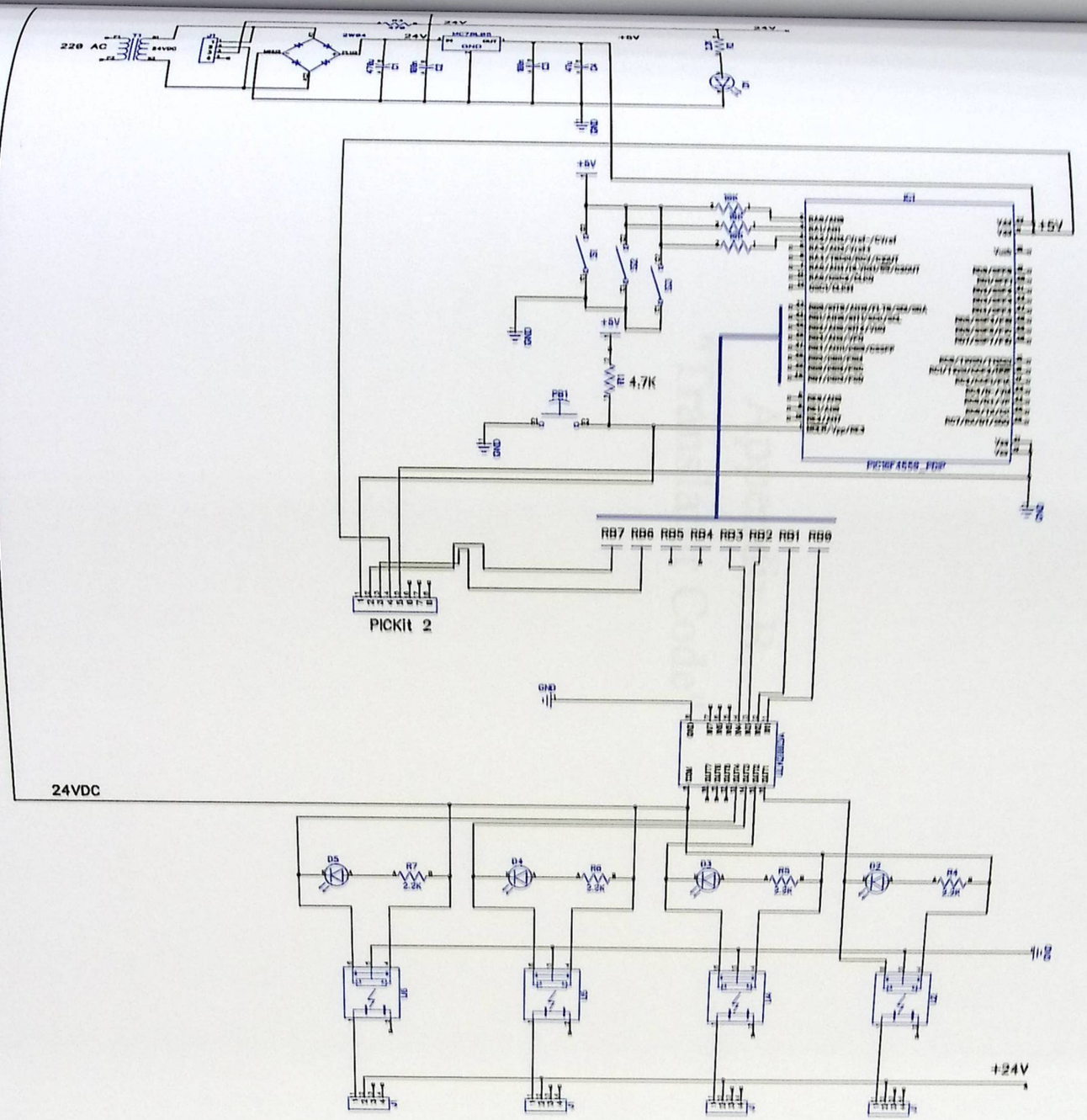
5. http://en.wikipedia.org/wiki/Contactor.

6. http://en.wikipedia.org/wiki/Relay.

7. http://www.switches.machinedesign.com/guiEdits/Content/bdeee4/bdeee4_17.aspx.

8. http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29

Appendix A
"Circuit Diagram"


220 AC
24V
+5V
4.7K
RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0
PICKit 2
24VDC
+24V
D5
D4
D3
D2
R7
R6
R5
R4
2.2K

Appendix B
"Translator Code"

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace GProject
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }


        public static string FindKey(Hashtable HT, string myValue)// Function to get
key from Value search
        {
            string myKey = "";

            foreach (string aKey in HT.Keys)
            {
                if (HT[aKey].ToString() == myValue)
                    myKey = aKey;
            }

            if (myKey == "")
                myKey = "No such that";

            return myKey;
        }



        public static string RetrieveS(Hashtable HR) // retrieve all keys in sereis
        {
            string r = "";
            int c = HR.Count - 1;
            int i = 0;

            foreach (string key in HR.Keys)
            {
                r += "" + key + " == 0 ";
                if (i != c)
                {
                    r += " && ";
                }
```

```
        else
            return r;

        i++;
    }

    return r;
}

public static string RetrieveP(Hashtable HR) // retrieve all keys in paralell
{
    string r = "";
    int c = HR.Count - 1;
    int i = 0;

    foreach (string key in HR.Keys)
    {
        r += "" + key + " == 0 ";
        if (i != c)
        {
            r += " || ";
        }

        else
            return r;

        i++;
    }

    return r;
}


private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog ofn = new OpenFileDialog();
    ofn.Filter = "Text Files (*.txt)|*.txt|All files (*.*)|*.*";
    ofn.Title = "Type File";

    if (ofn.ShowDialog() == DialogResult.Cancel)
        return;
    FileStream strm;
    try
    {
        strm = new FileStream(ofn.FileName, FileMode.Open, FileAccess.Read);
        StreamReader rdr = new StreamReader(strm);
        while (rdr.Peek() >= 0)
        {
            string str = rdr.ReadToEnd();
            textBox1.Text = str;

        }
    }
    catch (Exception)
```

```
        {
            MessageBox.Show("Error opening file", "File Error",
                            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }

    private void saveToolStripMenuItem_Click(object sender, EventArgs e)
    {
        SaveFileDialog sfile = new SaveFileDialog();
        sfile.Filter = "Text Files (*.txt)|*.txt";
        sfile.Title = "Save";

        try
        {
            sfile.ShowDialog();
            StreamWriter sf = new StreamWriter(sfile.FileName);
            sf.Write(textBox1.Text);
            sf.Close();
        }
        catch (Exception err)
        {
            MessageBox.Show(err.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void undoToolStripMenuItem_Click(object sender, EventArgs e)
    {
        textBox1.Undo();
    }

    private void cutToolStripMenuItem_Click(object sender, EventArgs e)
    {
        textBox1.Cut();
    }

    private void copyToolStripMenuItem_Click(object sender, EventArgs e)
    {
        textBox1.Copy();
    }

    Private void pasteToolStripMenuItem_Click(object sender, EventArgs e)
    {
        textBox1.Paste();
    }

    Private void deleteToolStripMenuItem_Click(object sender, EventArgs e)
    {
        textBox1.DeselectAll();
```

```
}

private void selectAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    textBox1.SelectAll();
}

private void openToolStripButton_Click(object sender, EventArgs e)
{
    OpenFileDialog ofn = new OpenFileDialog();
    ofn.Filter = "Text Files (*.txt)|*.txt|All files (*.*)|*.*";
    ofn.Title = "Type File";

    if (ofn.ShowDialog() == DialogResult.Cancel)
        return;
    FileStream strm;
    try
    {
        strm = new FileStream(ofn.FileName, FileMode.Open, FileAccess.Read);
        StreamReader rdr = new StreamReader(strm);
        while (rdr.Peek() >= 0)
        {
            string str = rdr.ReadToEnd();
            textBox1.Text = str;

        }
    }
    catch (Exception)
    {
        MessageBox.Show("Error opening file", "File Error",
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
}

private void saveToolStripButton_Click(object sender, EventArgs e)
{
    SaveFileDialog sfile = new SaveFileDialog();
    sfile.Filter = "Text Files (*.txt)|*.txt";
    sfile.Title = "Save";

    try
    {
        sfile.ShowDialog();
        StreamWriter sf = new StreamWriter(sfile.FileName);
        sf.Write(textBox1.Text);
        sf.Close();
    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}
```

```
private void cutToolStripButton_Click(object sender, EventArgs e)
{
    textBox1.Cut();
}

private void copyToolStripButton_Click(object sender, EventArgs e)
{
    textBox1.Copy();
}

private void pasteToolStripButton_Click(object sender, EventArgs e)
{
    textBox1.Paste();
}

private void button1_Click(object sender, EventArgs e)
{

    string ins; // instruction
    string label; // Label
    string op; // operand
    string sign; // any other sign
    string NET = ""; // instruction of network
    int ln = 0; // line number
    int c = 0; // counter
    int IC = 0; // Input counter
    int OC = 0; //  Output counter

    /* A-----I------0.1 */
    /* ins   label    ope */


    toolStripProgressBar1.Value = 0;

    Hashtable IPar = new Hashtable(); // Paralell input
    Hashtable OPar = new Hashtable(); // Paralell output

    Hashtable ISer = new Hashtable(); // Series input
    Hashtable OSer = new Hashtable(); // Sereis output

    SaveFileDialog cfile = new SaveFileDialog();

    cfile.Filter = "Text File (*.txt)|*.txt | C File (*.C)|*.C";
    cfile.Title = "Save";

    cfile.ShowDialog();
    StreamWriter cf = new StreamWriter(cfile.FileName);

    string[] line = new string[textBox1.Lines.Length];


    if (line.Length != 0)
    {
```

```
for (int count = 0; count < line.Length; count++)
{
    line = textBox1.Text.Split('\n');
}


cf.WriteLine("#include<p18f4550.h>");
cf.WriteLine("#pragma config WDT=OFF");
cf.WriteLine("#pragma config FOSC=INTOSC_HS");
cf.WriteLine("#pragma config LVP = OFF");
cf.WriteLine("void main(void)");
cf.WriteLine("{");
cf.WriteLine("int S = 0;");
cf.WriteLine("int R = 0;");
cf.WriteLine("OSCCON=OSCCON&0b11111111;");
cf.WriteLine("ADCON1=15;");
cf.WriteLine("TRISA = 255; // input port");
cf.WriteLine("TRISB = 0;  // output port");
cf.WriteLine("while(1)");
cf.WriteLine("{");

toolStripProgressBar1.Maximum = line.Length;


for (c =0 ; c < line.Length; c++)
{

    toolStripProgressBar1.Value = c;
    toolStripProgressBar1.Step = 10;

    ln = c;

Network:


    line[c] = line[c].Trim();
    ins = line[c].Substring(0, 1);
    sign = line[c].Substring(1, 1);
    label = op = "";
    if (line[c].Length > 5)
        label = line[c].Substring(6, 1);

    if (line[c].Length > 12)
        op = line[c].Substring(13, 3);

    if (line[c].Length > 3)
        NET = line[c].Substring(0, 3);

    if (NET == "Net" || NET == "net")
    {
        //ln = c;
        c++;
        ISer.Clear();
        IPar.Clear();
```

```
    OPar.Clear();
    NET = "";
    goto Network;
}


if (ins == "A" && sign == "(")
{
    c++;


    while (line.Length > c)
    {
        line[c] = line[c].Trim();
        ins = line[c].Substring(0, 1);
        label = op = "";

        if (line[c].Length > 5)
            label = line[c].Substring(6, 1);

        if (line[c].Length > 12)
            op = line[c].Substring(13, 3);

        if(line[c].Length > 1 )
            sign = line[c].Substring(1, 1);


        if (ins == "A" && label == "I")
        {
            ISer.Add("PORTAbits.RA" + IC, IC);
            IC++;

        }

        if (ins == "O")
        {
            IPar.Add("PORTAbits.RA" + IC, IC);
            IC++;
        }


        if (ins == "=")
        {
            OPar.Add("PORTBbits.RB" + OC, OC);
            OC++;
        }

        if (ins == "S" || ins == "R")
        {
            goto SR;
        }

        for (int m = 0; m < OPar.Count; m++)
        {
```

```
                    if (ISer.Count >= 1 || IPar.Count >= 1) //Multi Entity
                    {
                        if (ISer.Count == 1)
                        {
                            cf.WriteLine("if(" + RetrieveS(ISer) + " && " +
                        }
                        else
                        {
                            cf.WriteLine("if(" + RetrieveS(ISer) +
                        }
                        cf.WriteLine("{");
                        cf.WriteLine(FindKey(OPar, Convert.ToString(m)) +
                        cf.WriteLine("}");
                        m++;
                    }
                }
                c++;

                if (ins == ")")
                {
                    //c++;
                }

            }
            ISer.Clear();
            IPar.Clear();
            OPar.Clear();
            goto End1;
        }

    SR:

        if (ins == "A")
        {
            if (label == "I")
            {
                ISer.Add("PORTAbits.RA" + IC, IC);
                IC++;
            }
        }


        if (ins == "O")
        {
            IPar.Add("PORTAbits.RA" + IC, IC);
            IC++;
        }


        if (ins == "=")
```

```
{
    OPar.Add("PORTBbits.RB" + OC, OC);
    OC++;
}


if (ins == "S" || ins == "R") // SR
{

    //cf.WriteLine("S = 0;");
    //cf.WriteLine("R = 0;");

    ISer.Clear();
    IPar.Clear();

    for (int nc = 0 ; nc < line.Length ; nc++)
    {
        c = nc;

        toolStripProgressBar1.Value = nc;

        line[nc] = line[nc].Trim();

        ins = line[nc].Substring(0, 1);

        if (ins == "N")
        {
            c += 1;
            nc = c;
            line[nc] = line[nc].Trim();
            ISer.Clear();
            ins = line[nc].Substring(0, 1).Trim();
        }

        if (line[nc].Length > 1)
        sign = line[nc].Substring(1, 1).Trim();

        label = op = "";
        if (line[nc].Length > 5)
            label = line[nc].Substring(6, 1).Trim();

        if (line[nc].Length > 12)
            op = line[nc].Substring(13, 3).Trim();

        if (line[c].Length > 3)
            NET = line[nc].Substring(0, 3).Trim();

        if (NET == "Net" || NET == "net")
        {
            c++;
            ISer.Clear();
            IPar.Clear();
            OPar.Clear();
            NET = "";
```

```
        goto Network;
    }


    if (ins == "A" && label == "I")
    {
        ISer.Add("PORTAbits.RA" + nc, nc);
    }

    if (ins == "O" && label == "I")
    {
        IPar.Add("PORTAbits.RA" + nc, nc);
    }



    if (ins == "S")
    {
        cf.WriteLine("S = 0;");
        cf.WriteLine("R = 0;");

        OPar.Clear();
        OPar.Add("PORTBbits.RB0" , Convert.ToString(0));

        if (ISer.Count > 1)
        {
            cf.WriteLine("if(" + RetrieveS(ISer) + ")");
            cf.WriteLine("{");
            cf.WriteLine("S = 1 ;");
            cf.WriteLine("}");
        }

        if (IPar.Count > 1)
        {
            cf.WriteLine("if(" + RetrieveP(IPar) + ")");
            cf.WriteLine("{");
            cf.WriteLine("S = 1 ;");
            cf.WriteLine("}");
        }

        if (ISer.Count == 1 || IPar.Count == 1)
        {
            cf.WriteLine("if(" + RetrieveS(ISer) + ")");
            cf.WriteLine("{");
            cf.WriteLine("S = 1 ;");
            cf.WriteLine("}");
        }

        ISer.Clear();
        IPar.Clear();
        OPar.Clear();
    }

    if (ins == "R")
```

```
        {
            OPar.Clear();
            OPar.Add("PORTBbits.RB0", Convert.ToString(0));

            if (ISer.Count > 1)
            {
                cf.WriteLine("if(" + RetrieveS(ISer) + ")");
                cf.WriteLine("{");
                cf.WriteLine("R = 1 ;");
                cf.WriteLine("}");
            }

            if (IPar.Count > 1)
            {
                cf.WriteLine("if(" + RetrieveP(IPar) + ")");
                cf.WriteLine("{");
                cf.WriteLine("R = 1 ;");
                cf.WriteLine("}");
            }

            if (ISer.Count == 1 || IPar.Count == 1)
            {
                cf.WriteLine("if(" + RetrieveS(ISer) + ")");
                cf.WriteLine("{");
                cf.WriteLine("R = 1 ;");
                cf.WriteLine("}");
            }

            cf.WriteLine("if( S == 0 && R == 1 )");
            cf.WriteLine("{");
            cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
");

            cf.WriteLine("}");

            cf.WriteLine("if( S == 1 && R == 0 )");
            cf.WriteLine("{");
            cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
");

            cf.WriteLine("}");

            cf.WriteLine("if( S == 1 && R == 1 )");
            cf.WriteLine("{");
            cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
");

            cf.WriteLine("}");

        }

    }
    ISer.Clear();
    IPar.Clear();
    OPar.Clear();

}
```

```
                {
                    OPar.Clear();
                    OPar.Add("PORTBbits.RB0", Convert.ToString(0));

                    if (ISer.Count > 1)
                    {
                        cf.WriteLine("if(" + RetrieveS(ISer) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine("R = 1 ;");
                        cf.WriteLine("}");
                    }

                    if (IPar.Count > 1)
                    {
                        cf.WriteLine("if(" + RetrieveP(IPar) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine("R = 1 ;");
                        cf.WriteLine("}");
                    }

                    if (ISer.Count == 1 || IPar.Count == 1)
                    {
                        cf.WriteLine("if(" + RetrieveS(ISer) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine("R = 1 ;");
                        cf.WriteLine("}");
                    }

                    cf.WriteLine("if( S == 0 && R == 1 )");
                    cf.WriteLine("{");
                    cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
0 ;");

                    cf.WriteLine("}");

                    cf.WriteLine("if( S == 1 && R == 0 )");
                    cf.WriteLine("{");
                    cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
1 ;");

                    cf.WriteLine("}");

                    cf.WriteLine("if( S == 1 && R == 1 )");
                    cf.WriteLine("{");
                    cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
0 ;");

                    cf.WriteLine("}");

                }

            }
            ISer.Clear();
            IPar.Clear();
            OPar.Clear();

        }
```

```
                {
                    OPar.Clear();
                    OPar.Add("PORTBbits.RB0", Convert.ToString(0));

                    if (ISer.Count > 1)
                    {
                        cf.WriteLine("if(" + RetrieveS(ISer) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine("R = 1 ;");
                        cf.WriteLine("}");
                    }

                    if (IPar.Count > 1)
                    {
                        cf.WriteLine("if(" + RetrieveP(IPar) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine("R = 1 ;");
                        cf.WriteLine("}");
                    }

                    if (ISer.Count == 1 || IPar.Count == 1)
                    {
                        cf.WriteLine("if(" + RetrieveS(ISer) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine("R = 1 ;");
                        cf.WriteLine("}");
                    }

                    cf.WriteLine("if( S == 0 && R == 1 )");
                    cf.WriteLine("{");
                    cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
0 ;");

                    cf.WriteLine("}");

                    cf.WriteLine("if( S == 1 && R == 0 )");
                    cf.WriteLine("{");
                    cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
1 ;");

                    cf.WriteLine("}");

                    cf.WriteLine("if( S == 1 && R == 1 )");
                    cf.WriteLine("{");
                    cf.WriteLine(FindKey(OPar, Convert.ToString(0)) + " =
0 ;");

                    cf.WriteLine("}");

                }

            }
            ISer.Clear();
            IPar.Clear();
            OPar.Clear();

        }
```

```
                for (int p = 0; p < OPar.Count; p++)
                {

                    if (ISer.Count == 1) // One Entity One-To-One in series
                    {
                        cf.WriteLine("if(" + FindKey(ISer, Convert.ToString(p)) +
"==" + "0" + ")");
                        cf.WriteLine("{");
                        cf.WriteLine(FindKey(OPar, Convert.ToString(p)) + "=" + "1"
+ ";");
                        cf.WriteLine("}");
                    }

                    if (IPar.Count == 1) // One Entity One-To-One in paralell
                    {
                        cf.WriteLine("if(" + FindKey(IPar, Convert.ToString(p)) +
"==" + "0" + ")");
                        cf.WriteLine("{");
                        cf.WriteLine(FindKey(OPar, Convert.ToString(p)) + "=" + "1"
+ ";");
                        cf.WriteLine("}");
                    }

                    if (ISer.Count > 1) //Multi Entity in sereis
                    {
                        cf.WriteLine("if(" + RetrieveS(ISer) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine(FindKey(OPar, Convert.ToString(p)) + "=" + "1"
+ ";");
                        cf.WriteLine("}");
                    }

                    if (IPar.Count > 1) //Multi Entity iin paralell
                    {
                        cf.WriteLine("if(" + RetrieveP(IPar) + ")");
                        cf.WriteLine("{");
                        cf.WriteLine(FindKey(OPar, Convert.ToString(p)) + "=" + "1"
+ ";");
                        cf.WriteLine("}");
                    }

                }

            }
        }

        else
        {
            MessageBox.Show("This File is Empty!", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
```

```
    End1:
        cf.WriteLine("} ");
        cf.WriteLine("} ");


                cf.Close();


            }

    private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Help h = new Help();
        h.Show();
    }


        }
}
```

Appendix C
"Statement List Instruction"

Siemens S7 Statement List (STL)

sorted alphabetically

Mnemonic	Description
)	Nesting Closed
+	Add Integer Constant (16, 32-Bit)
+AR1	Add ACC1 to Address Register 1
+AR2	Add ACC1 to Address Register 2
+I +D +R	Add ACC1 and ACC2
-I -D -R	Subtract ACC1 from ACC2
*I *D *R	Multiply ACC1 and ACC2
/I /D /R	Divide ACC2 by ACC1
=	Assign
==I ==D ==R	ACC2 is equal to ACC1
<=I <=D <=R	ACC2 is less then equal to ACC1
<>I <>D <>R	ACC2 is not equal to ACC1
<I <D <R	ACC2 is less then to ACC1
>=I >=D >=R	ACC2 is greater then equal to ACC1
>I >D >R	ACC2 is greater then to ACC1
A	And
A(	And with Nesting Open
ABS	Absolute Value
ACOS	Arc Cosine
AD	AND Double Word
AN	And Not
AN(	And Not with Nesting Open
ASIN	Arc Sine
ATAN	Arc Tangent
AW	AND Word
BE	Block End
BEC	Block End Conditional
BEU	Block End Unconditional
BLD	Program Display Instruction (Null)
BTD	BCD to Integer
BTI	BCD to Integer
CAD	Change Byte Sequence in ACC1 Double
CALL	Call FC,FB,SFC,SFB
CAR	Exchange Address Register 1 with Address Register 2
CAW	Change Byte Sequence in ACC1 Word
CC	Conditional Call
CD	Counter Down
CDB	Exchange Shared DB and Instance DB
CLR	Clear RLO (=0)
COS	Cosine of Angles
CU	Counter Up
DEC	Decrement ACC
DTB	Double Integer to BCD
DTR	Double Integer to Floating-Point
ENT	Enter ACC Stack
EXP	Exponential Value
FN	Edge Negative
FP	Edge Positive
FR	Enable Timer/Counter (Free)
INC	Increment ACC
INVD	Ones Complement Double Integer
INVI	Ones Complement Integer
ITB	Integer to BCD
ITD	Integer to Double Integer
JBI	Jump if BR = 1
JC	Jump if RLO = 1
JCB	Jump if RLO = 1 with BR
JCN	Jump if RLO = 0
JL	Jump to Labels
JM	Jump if Minus
JMZ	Jump if Minus or Zero
JN	Jump if Not Zero
JNB	Jump if RLO = 0 with BR
JNBI	Jump if BR = 0
JO	Jump if OV = 1
JOS	Jump if OS = 1
JP	Jump if Plus
JPZ	Jump if Plus or Zero
JU	Jump Unconditional
JUO	Jump if Unordered
JZ	Jump if Zero
L	Load
L	Load Current Timer/Counter Value into ACC1 as Integer (i.e. L T 32)
L DBLG	Load Length of Shared DB in ACC1
L DBNO	Load Number of Shared DB in ACC1
L DILG	Load Length of Instance DB in ACC1
L DINO	Load Number of Instance DB in ACC1
L STW	Load Status Word into ACC1
LAR1	Load Address Register 1 from ACC1
LAR1 <D>	Load Address Register 1 with Double Integer (32-Bit Pointer)
LAR1 AR2	Load Address Register 1 from Address Register 2
LAR2	Load Address Register 2 from ACC1
LAR2 <D>	Load Address Register 2 with Double Integer (32-Bit Pointer)
LC	Load Current Timer/Counter Value into ACC1 as BCD (i.e. LC T 32)
LEAVE	Leave ACC Stack
LN	Natural Logarithm
LOOP	Loop

Siemens S7 Statement List (STL)

sorted alphabetically

Mnemonic	Description
)	Nesting Closed
+	Add Integer Constant (16, 32-Bit)
+AR1	Add ACC1 to Address Register 1
+AR2	Add ACC1 to Address Register 2
+I +D +R	Add ACC1 and ACC2
-I -D -R	Subtract ACC1 from ACC2
*I *D *R	Multiply ACC1 and ACC2
/I /D /R	Divide ACC2 by ACC1
=	Assign
==I ==D ==R	ACC2 is equal to ACC1
<=I <=D <=R	ACC2 is less then equal to ACC1
<>I <>D <>R	ACC2 is not equal to ACC1
<I <D <R	ACC2 is less then to ACC1
>=I >=D >=R	ACC2 is greater then equal to ACC1
>I >D >R	ACC2 is greater then to ACC1
A	And
A(	And with Nesting Open
ABS	Absolute Value
ACOS	Arc Cosine
AD	AND Double Word
AN	And Not
AN(	And Not with Nesting Open
ASIN	Arc Sine
ATAN	Arc Tangent
AW	AND Word
BE	Block End
BEC	Block End Conditional
BEU	Block End Unconditional
BLD	Program Display Instruction (Null)
BTD	BCD to Integer
BTI	BCD to Integer
CAD	Change Byte Sequence in ACC1 Double
CALL	Call FC,FB,SFC,SFB
CAR	Exchange Address Register 1 with Address Register 2
CAW	Change Byte Sequence in ACC1 Word
CC	Conditional Call
CD	Counter Down
CDB	Exchange Shared DB and Instance DB
CLR	Clear RLO (=0)
COS	Cosine of Angles
CU	Counter Up
DEC	Decrement ACC
DTB	Double Integer to BCD
DTR	Double Integer to Floating-Point
ENT	Enter ACC Stack
EXP	Exponential Value
FN	Edge Negative
FP	Edge Positive
FR	Enable Timer/Counter (Free)
INC	Increment ACC
INVD	Ones Complement Double Integer
INVI	Ones Complement Integer
ITB	Integer to BCD
ITD	Integer to Double Integer
JBI	Jump if BR = 1
JC	Jump if RLO = 1
JCB	Jump if RLO = 1 with BR
JCN	Jump if RLO = 0
JL	Jump to Labels
JM	Jump if Minus
JMZ	Jump if Minus or Zero
JN	Jump if Not Zero
JNB	Jump if RLO = 0 with BR
JNBI	Jump if BR = 0
JO	Jump if OV = 1
JOS	Jump if OS = 1
JP	Jump if Plus
JPZ	Jump if Plus or Zero
JU	Jump Unconditional
JUO	Jump if Unordered
JZ	Jump if Zero
L	Load
L	Load Current Timer/Counter Value into ACC1 as Integer (i.e. L T 32)
L DBLG	Load Length of Shared DB in ACC1
L DBNO	Load Number of Shared DB in ACC1
L DILG	Load Length of Instance DB in ACC1
L DINO	Load Number of Instance DB in ACC1
L STW	Load Status Word into ACC1
LAR1	Load Address Register 1 from ACC1
LAR1 <D>	Load Address Register 1 with Double Integer (32-Bit Pointer)
LAR1 AR2	Load Address Register 1 from Address Register 2
LAR2	Load Address Register 2 from ACC1
LAR2 <D>	Load Address Register 2 with Double Integer (32-Bit Pointer)
LC	Load Current Timer/Counter Value into ACC1 as BCD (i.e. LC T 32)
LEAVE	Leave ACC Stack
LN	Natural Logarithm
LOOP	Loop

Mnemonic	Description
MCR(	Save RLO in MCR Stack, Begin MCR
)MCR	End MCR
MCRA	Activate MCR
MCRD	Deactivate MCR
MOD	Division Remainder Double Integer
NEGD	Twos Complement Double Integer
NEGI	Twos Complement Integer
NEGR	Negate Floating-Point Number
NOP 0	Null Instruction
NOP 1	Null Instruction
NOT	Negate RLO
O	Or
O(	Or with Nesting Open
OD	OR Double Word
ON	Or Not
ON(	Or Not with Nesting Open
OPN	Open a Data Block
OW	OR Word
POP	Pop accumulators
PUSH	Push accumulators
R	Reset
R	Reset Timer/Counter Value (i.e. R T 32)
RLD	Rotate Left Double Word
RLDA	Rotate ACC1 Left via CC 1
RND	Round
RND-	Round to Lower Double Integer
RND+	Round to Upper Double Integer
RRD	Rotate Right Double Word
RRDA	Rotate ACC1 Right via CC 1
S	Set
S	Set Counter Preset Value (i.e. S C 15)
SAVE	Save RLO in BR Register
SD	On-Delay Timer
SE	Extended Pulse Timer

Mnemonic	Description
SET	Set RLO (=1)
SF	Off-Delay Timer
SIN	Sine of Angles
SLD	Shift Left Double Word
SLW	Shift Left Word
SP	Pulse Timer
SQR	Square
SQRT	Square Root
SRD	Shift Right Double Word
SRW	Shift Right Word
SS	Retentive On-Delay Timer
SSD	Shift Sign Double Integer
SSI	Shift Sign Integer
T	Transfer
T STW	Transfer ACC1 into Status Word
TAK	Toggle ACC1 with ACC2
TAN	Tangent of Angles
TAR1	Transfer Address Register 1 to ACC1
TAR1 <D>	Transfer Address Register 1 to Destination (32-Bit Pointer)
TAR1 AR2	Transfer Address Register 1 to Address Register 2
TAR2	Transfer Address Register 2 to ACC1
TAR2 <D>	Transfer Address Register 2 to Destination (32-Bit Pointer)
TRUNC	Truncate
UC	Unconditional Call
X	Exclusive Or
X(	Exclusive Or with Nesting Open
XN	Exclusive Or Not
XN(	Exclusive Or Not with Nesting Open
XOD	Exclusive Or Double Word
XOW	Exclusive Or Word

Formats	
B#	Byte (8 bit)
W#	Word (16 bit)
L#	Long (32 bit)
S5Time#	S5 Time (2H46M30S0MS)
T#	IEC Time (24D20H31M23S648MS)
D#	IEC Date (2007-10-28)
TOD#	Time of Day (23:59:59.999)
C#	BCD
P#	Pointer Address
2#	Binary
16#	Hexadecimal
#Symbol	Local stack variable
//	Comment

OBs	
1	Main Program Scan
10-17	Time of Day
20-23	Time Delay
30-38	Cyclic (Periodic)
40-47	Hardware
80	Time Error
81	Power Supply Error
82	Diagnostic Interrupt
83	Insert/Remove Module Interrupt
84	CPU Hardware Fault
85	Program Cycle Error
86	Rack Failure - Missing Profibus device
87	Communication Error
100	Warm restart
101	Hot restart
102	Cold restart
121	Programming Error
122	I/O Access Error

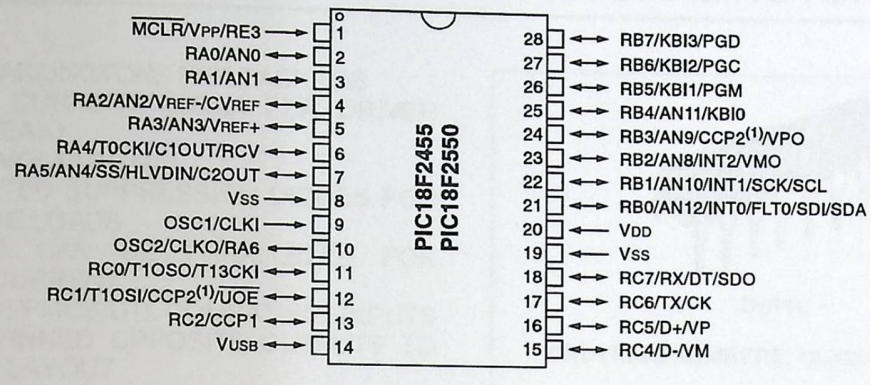
PLC dev

www.plcdev.com

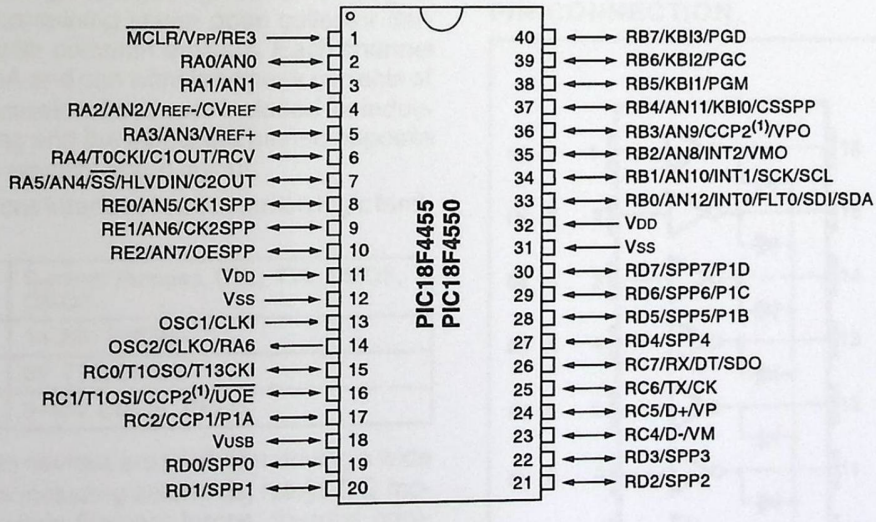
Appendix D
"Data Sheets"

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP

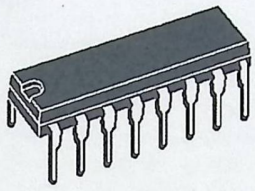


Note 1: RB3 is the alternate pin for CCP2 multiplexing.

ULN2001A-ULN2002A
ULN2003A-ULN2004A

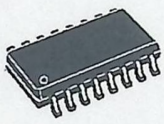
SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT



DIP16

ORDERING NUMBERS: ULN2001A/2A/3A/4A



SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

DESCRIPTION

The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

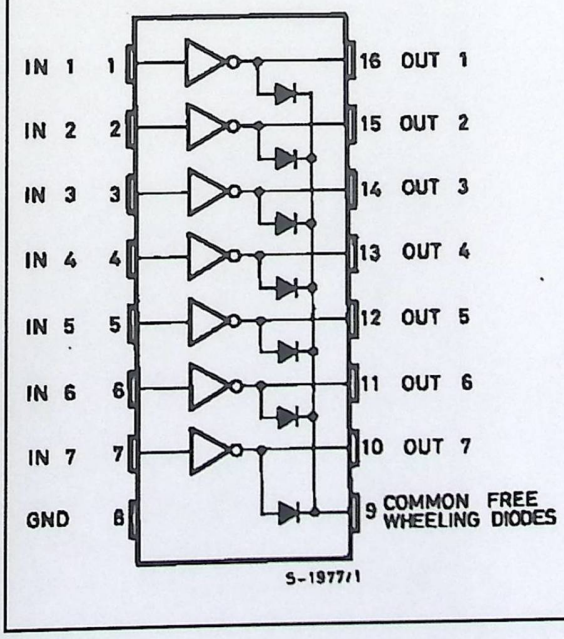
The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6–15V CMOS, PMOS

These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003Aand 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.

PIN CONNECTION



September 1998

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

MC78XX/LM78XX/MC78XXA

3-Terminal 1A Positive Voltage Regulator

Features

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The MC78XX/LM78XX/MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

TO-220

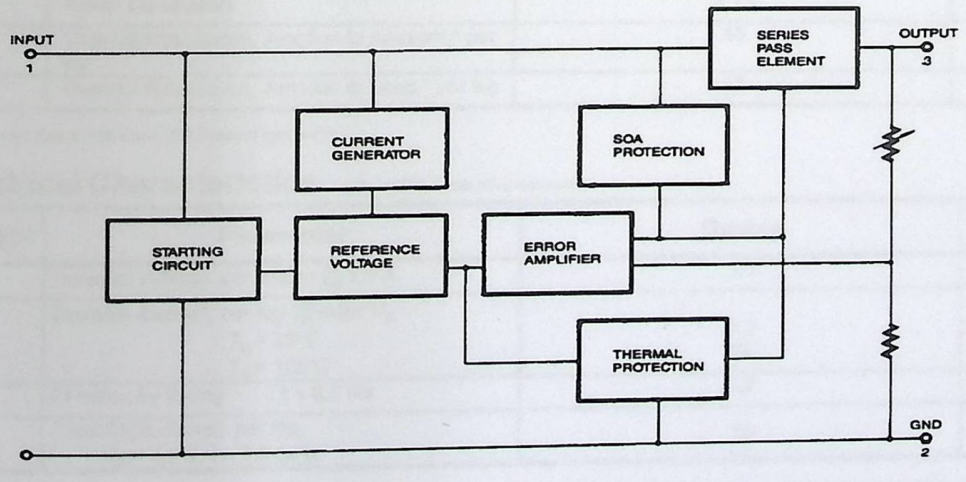
1

D-PAK

1

1. Input 2. GND 3. Output

Internal Block Digram



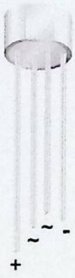
Rev. 1.0.1

FAIRCHILD
SEMICONDUCTOR®

2W005G - 2W10G

Features

- Glass passivated junction.
- Ideal for printed circuit board.
- Reliable low cost construction technique results in inexpensive product.
- High surge current capability.
- UL certified, UL #E96005.



WOB

Bridge Rectifiers (Glass Passivated)

Absolute Maximum Ratings*

T_A = 25°C unless otherwise noted

Symbol	Parameter	Value							Units
		005G	01G	02G	04G	06G	08G	10G	
V_{RRM}	Maximum Repetitive Reverse Voltage	50	100	200	400	600	800	1000	V
V_{RMS}	Maximum RMS Bridge Input Voltage	35	70	140	280	420	560	700	V
V_R	DC Reverse Voltage (Rated V_R)	50	100	200	400	600	800	1000	V
$I_{F(AV)}$	Average Rectified Forward Current, @ T_A = 50°C	2.0							A
I_{FSM}	Non-repetitive Peak Forward Surge Current 8.3 ms Single Half-Sine-Wave	60							A
T_{stg}	Storage Temperature Range	-55 to +150							°C
T_J	Operating Junction Temperature	-55 to +150							°C

*These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

Thermal Characteristics

Symbol	Parameter	Value	Units
P_D	Power Dissipation	3.13	W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient,* per leg	40	°C/W
$R_{\theta JL}$	Thermal Resistance, Junction to Lead,* per leg	15	°C/W

*Device mounted on PCB with 0.375" (9.5 mm) lead length.

Electrical Characteristics

T_A = 25°C unless otherwise noted

Symbol	Parameter	Device	Units
V_F	Forward Voltage, per bridge @ 2.0 A	1.1	V
I_R	Reverse Current, per leg @ rated V_R T_A = 25°C T_A = 125°C	 5.0 500	 µA µA
	I^2t rating for fusing t < 8.3 ms	10	A^2s
C_T	Total Capacitance, per leg V_R = 4.0 V, f = 1.0 MHz	19	pF