ANNA UNIVERSITY OF TECHNOLOGY
COIMBATORE 641 047

BONAFIDE CERTIFICATE

P-4166

Certified that this project report titled "**AUTOMATION IN QUALITY CONTROL OF RELAYS USING LabVIEW**" is the bonafide work of

NISHANTH S	**0710106029**
PASIL RAJ R	**0710106031**
PREMKUMAR C M	**0710106036**
THARIQ AHAMED M	**0710106051**

who carried out the project under my supervision.

SIGNATURE
Prof.R.Annamalai, M.E
HEAD OF THE DEPARTMENT

Department of Electronics
& Instrumentation Engineering,
Kumaraguru College
of Technology.

SIGNATURE
Prof.R.Annamalai, M. E
SUPERVISOR
Head of the Department
Department of Electronics
&Instrumentation Engineering,
Kumaraguru College
of Technology.

The candidates were examined by us in the project viva-voice examination held on 18.04.2011

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We are greatly indebted to our beloved Principal **Dr. S. Ramachandran** who has been the backbone of all our deeds.

We profusely thank **Prof. R. Annamalai**, Head of the Department, Department of Electronics and Instrumentation Engineering, for lending a helping hand in this project.

We profusely thank our beloved guide **Mr.B.Mahendran**, Senior Manager – Works and **Mr.D.Rajanayagam**, Senior Manager – Standards and Systems, Salzer Electronics Ltd, Coimbatore – 641047, who helped us with excellent guidance and valuable suggestions throughout the project.

We are highly grateful to our beloved Project Coordinator, **Mr.S.Arun Jayakar**, Senior Lecturer, Project Guide **Prof.R.Annamalai**, HOD and **Ms.B.Veena Abirami**, Senior Lecturer Electronics and Instrumentation Engineering Department for their valuable guidance, timely help, constant encouragement and advice rendered throughout the project period for the successful completion of the same.

We are grateful to all the faculty members of the Department of Electronics and Instrumentation Engineering who have helped us in numerable ways.

We also thank our parents without whom we couldn't have come so far and friends for their timely help that culminated as good in the end.

ZER ELECTRONICS LTD.
ttipalayam, Coimbatore - 641 047, India.

April 12, 2011

CERTIFICATE

This is to certify that the following students of B.E.(EIE) Final Year students of **KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE ,641 006,** has successfully completed their Project Work entitled **"AUTOMATION OF QUALITY CONTROL OF RELAYS USING LabVIEW"** as a part of their course in our company from **Dec 2010 to March 2011.**

1. **Mr.NISHANTH.S (07BE129)**
2. **Mr.PASILRAJ.R (07BE131)**
3. **Mr.PREMKUMAR.C.M (07BE136)**
4. **Mr.THARIQ AHAMED.M (07BE151)**

They has evinced keen interest in absorbing the nature, concept and functions of our organization and their conduct and character were **good** during the period.

For **SALZER ELECTRONICS LIMITED**

(K.RAMAN)
ASSISTANT MANAGER – HR

ABSTRACT

ABSTRACT

In today's fast moving world, technology is improved to such an extent that every task performed by human beings is being automated with engineering solutions. One such engineering solution is the advent of Sensors and Controllers. Controllers are usually used for sensing input from the real world and controlling devices based on that input.

Manual operations in industries have been shifted to automation sector to reduce the workload of human labours. Looking in this perspective here in this project the relay testing procedure is automated to increase the efficiency and quality if the relays that are being manufactured.

Control of the entire system may be coordinated by a microcontroller or using graphical softwares like LabVIEW. Here the microcontroller provides the control whereas the graphical programming software provides the monitoring of the entire system.

LIST OF FIGURES

TABLE OF CONTENTS

1.1 NEED OF THE PROJECT:

In Relay manufacturing industries, a large human workforce is employed to test the quality of relays in order to avoid the manufacturing failures in relays. But it consumes a large amount of time to do the testing process which in turn reduces the production rate. Implementation of automation in quality testing of relays will save more time. As a result, we get high quality of relays. Also there will be a reasonable increase in the production rate of relay manufacturing industries.

1.2 OBJECTIVE OF THE PROJECT:

The main objective of this project is to automate the quality control of relays in relay manufacturing industries. This involves the following measurements:

- ✓ Measurement of voltage across the relay.
- ✓ Measurement of temperature across the relay.
- ✓ Lifespan of a particular relay.

Thus using the above measured parameters the quality of relay is evaluated.

1.3 OVERALL BLOCK DIAGRAM:

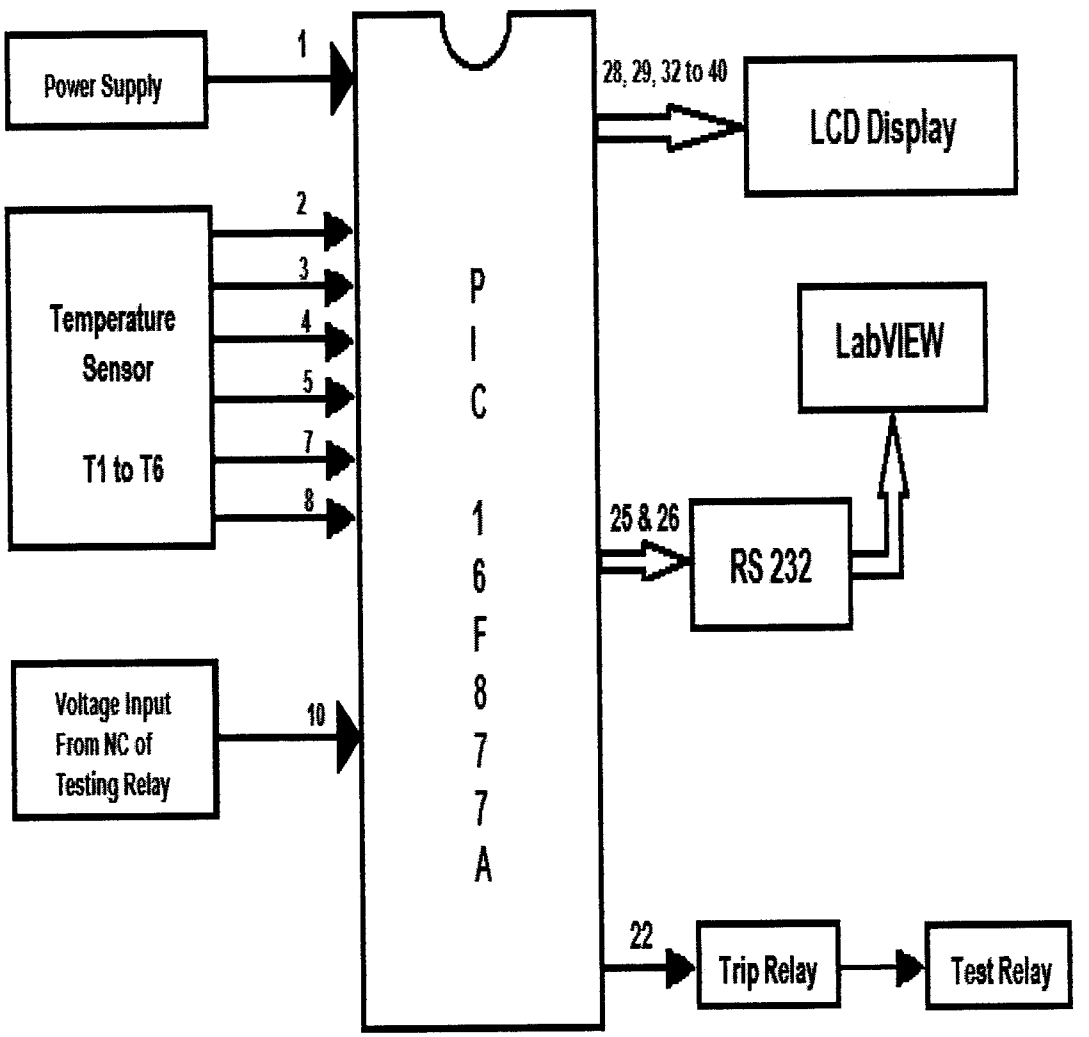


Fig 1.1 Block Diagram of Relay Quality Testing System

1.4 OVERALL CIRCUIIT DIAGRAM

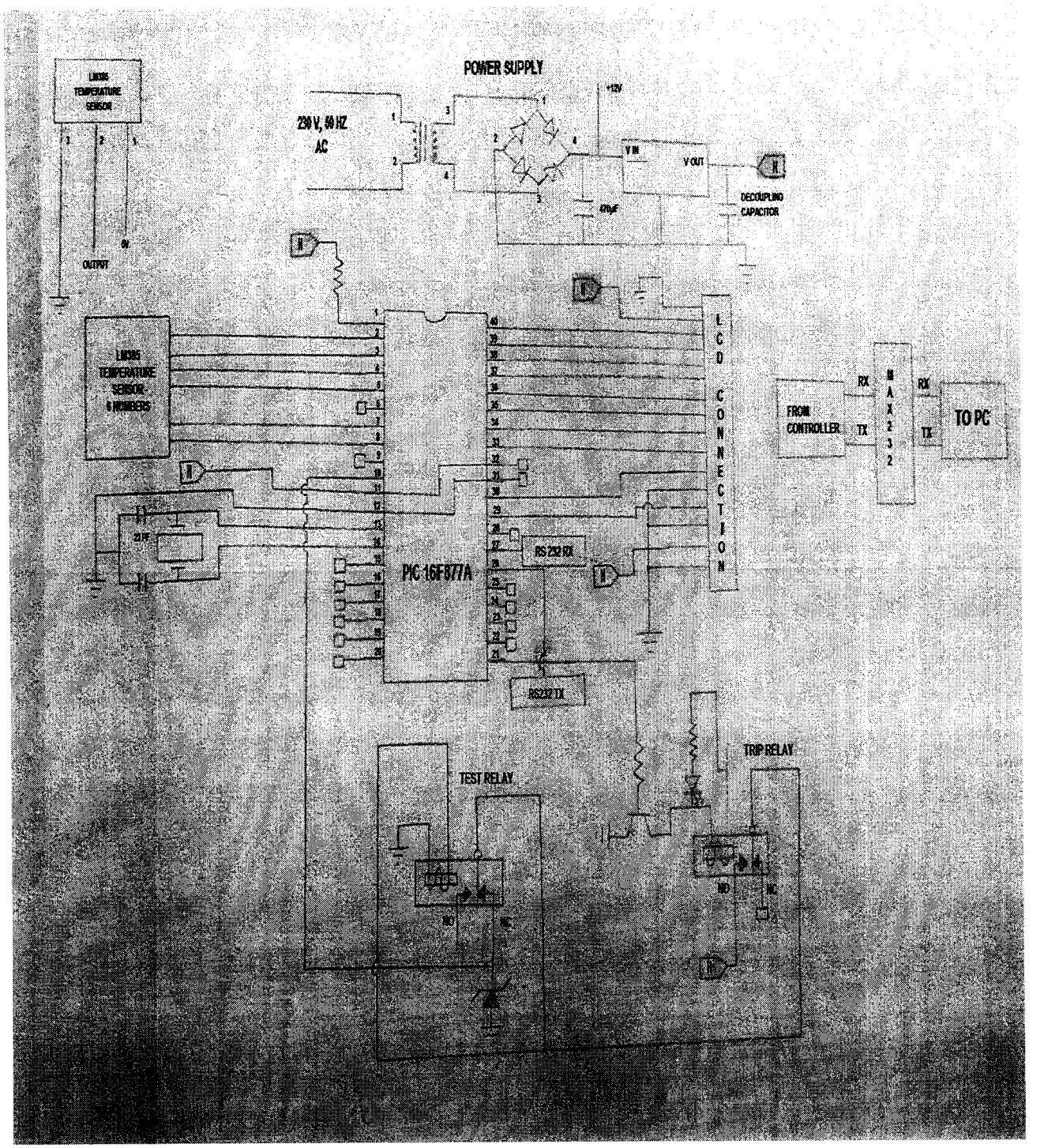


Fig 1.2 Circuit Diagram of Relay Quality Testing System

1.5 OVERVIEW OF THE COMPONENTS USED:

1.5.1 POWER SUPPLY:

The power supply circuit is built using filters, rectifiers, and voltage regulators. Starting with an ac voltage, a steady dc voltage is obtained by rectifying the ac voltage, then filtering to a dc level, and finally, regulated to obtain a desired fixed dc voltage. The regulated voltage is usually obtained from an IC voltage regulator unit, which remains the same even if the input dc voltage varies, or the output load connected to the dc voltage changes.

A block diagram of a typical power supply is shown in fig 1.3. The ac voltage, typically 120 V rms, is connected to a transformer, which steps that ac voltage down to the level for the desired dc output. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage. This resulting dc voltage usually has some ripple or ac voltage variation. A regulator circuit can use this dc input to provide a dc voltage that not only has much less ripple voltage but also remains the same dc value even if the input dc voltage varies somewhat, or the load connected to the output dc voltage changes. This voltage regulation is usually obtained using one of a number of popular voltage regulator IC units.

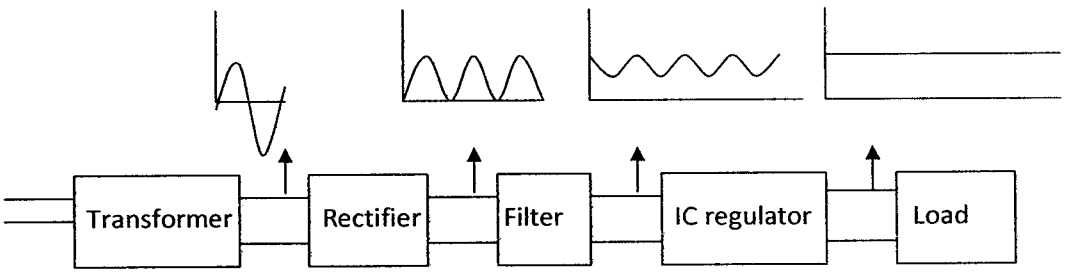


Fig 1.3 Block Diagram of Power Supply Unit

1.5.2 RELAYS:

A relay is an electrical switch that uses an electromagnet to move the switch from the off to on position instead of a person moving the switch. It takes a relatively small amount of power to turn on a relay but the relay can control something that draws much more power.

1.5.3 SENSORS:

LM35 series thermocouple is used for the temperature measurement across the relay terminals. Voltage input from the NC point of the relay is given to microcontroller for measuring voltage across it.

1.5.4 MICROCONTROLLER;

PIC 16F877A series microcontroller is used for data processing and controlling purposes. Data from real world are converted to digital levels using ADC section of the microcontroller.

1.5.5 RS 232 COMMUNICATION:

The data is transmitted and received between the microcontroller and PC or other device vice versa using RS232 communication.MAX 232 is used as the driver for RS232 communication.

CHAPTER 2

RELAY

2.1 INTRODUCTION

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever (common terminal) to oscillate between NO and NC of the relay. Relays allow one circuit to switch a second circuit which can be completely separate from the first. For example a low voltage battery circuit can use a relay to switch a 230V AC mains circuit. There is no electrical connection inside the relay between the two circuits; the link is magnetic and mechanical.

The coil of a relay passes a relatively large current, typically 30mA for a 12V relay, but it can be as much as 100mA for relays designed to operate from lower voltages. Most ICs (chips) cannot provide this current and a transistor is usually used to amplify the small IC current to the larger value required for the relay coil. The maximum output current for the popular 555 timer IC is 200mA so these devices can supply relay coils directly without amplification.

The bottom view and top view of the 5A/12V DC relay is shown in the below figure.

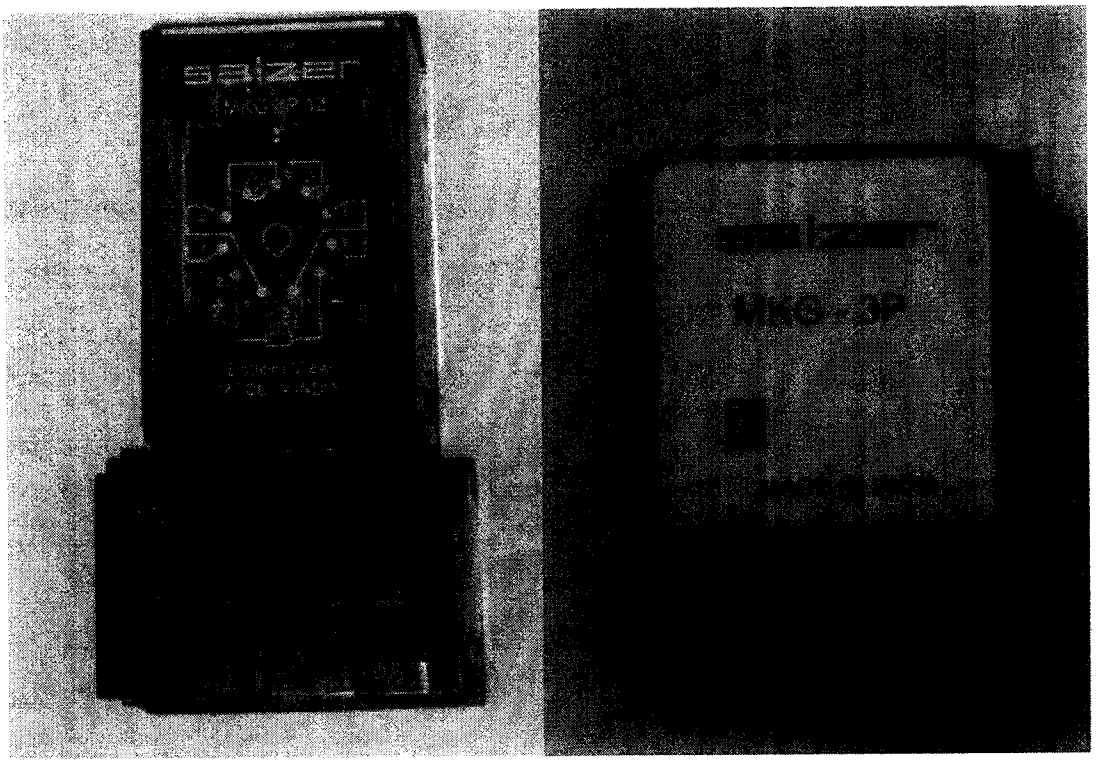


Fig 2.1 RELAY

RELAY BASICS:

A relay is an electrical switch that uses an electromagnet to move the switch from the off to on position instead of a person moving the switch. It takes a relatively small amount of power to turn on a relay but the relay can control something that draws much more power. Ex: A relay is used to control the air conditioner in your home. The AC unit probably runs off of 220VAC at around 30A. That's 6600 Watts! The coil that controls the relay may only need a few watts to pull the contacts together.

This is the schematic representation of a relay. The contacts at the top are normally open (i.e. not connected). When current is passed through the coil it

creates a magnetic field that pulls the switch closed (i.e. connects the top contacts). Usually a spring will pull the switch open again once the power is removed from the coil.

Relays are usually SPDT or DPDT but they can have many more sets of switch contacts, for example relays with 4 sets of changeover contacts are readily available. The animated picture shows a working relay with its coil and switch contacts. You can see a lever on the left being attracted by magnetism when the coil is switched on. This lever moves the switch contacts. There is one set of contacts (SPDT) in the foreground and another behind them, making the relay DPDT.

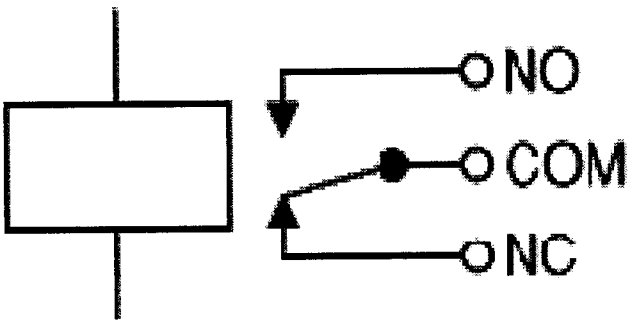


Fig 2.2 WORKING OF RELAY

The relay's switch connections are usually labeled COM, NC and NO:

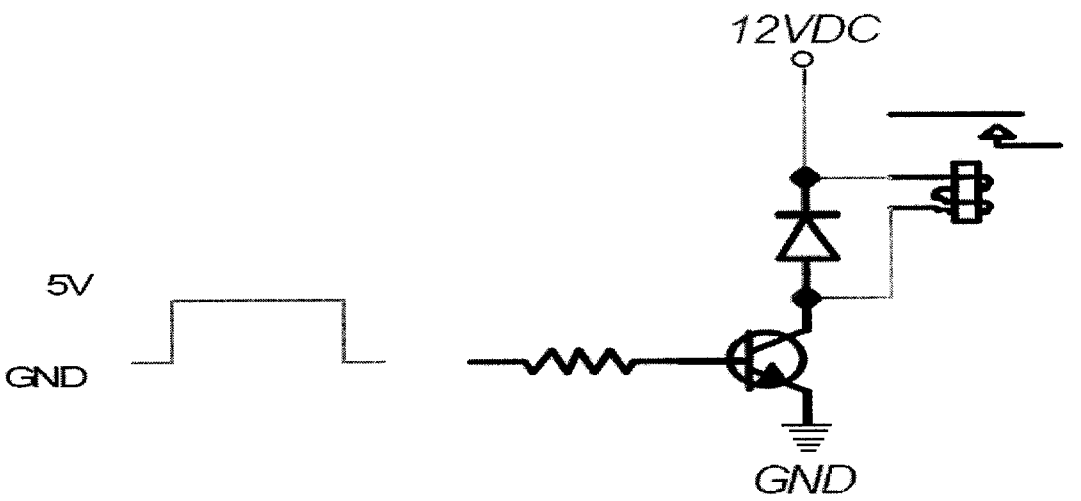
- **COM** = Common, always connect to this. It is the moving part of the switch.
- **NC** = Normally Closed, COM is connected to this when the relay coil is **off**.
- **NO** = Normally Open, COM is connected to this when the relay coil is **on**

PRACTICAL CONSIDERATIONS:

Check the relay datasheet for the expected lifetime (i.e. # of times it can open and close before failure). If the relay won't be used much (say to control the headlights on a car) a 20,000 cycle lifetime would last about 18 years if used three times a day. If the same relay was used to control a home AC unit which switches on and off much more often it would wear out in a few years. Some relays have lifetimes of over a million cycles.

Flyback Diode:

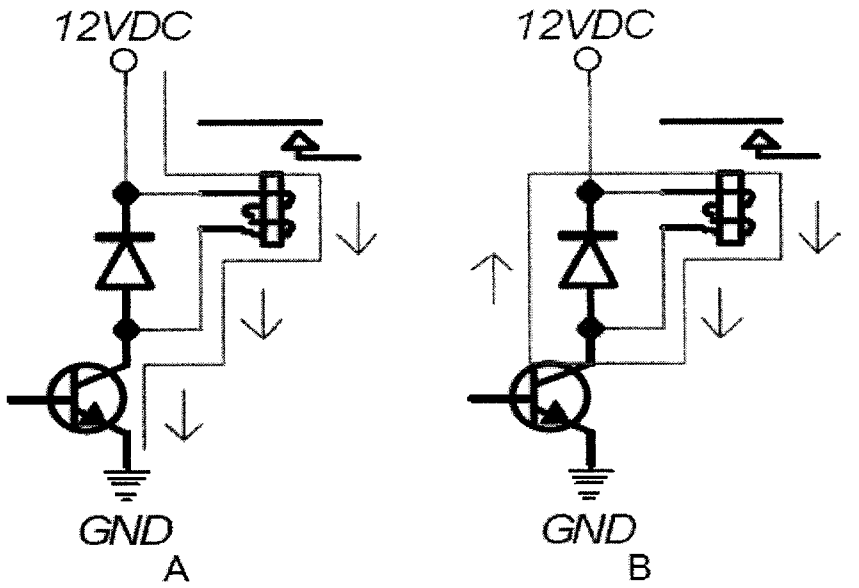
A relay coil is not only an electromagnet but it's also an inductor. When power is applied to the coil the current in the coil builds up and levels off at its rated current (depends on the DC resistance of the coil, I = V/R). Some energy is now stored in the coil's magnetic field ($E = 05LI^2$). When the current in the coil is turned off this stored energy has to go somewhere. The voltage across the coil quickly increase trying to keep the current in the coil flowing in the same direction (V = Ldi/dt). This voltage spike can reach hundreds or thousands of volts and can damage electronic parts.



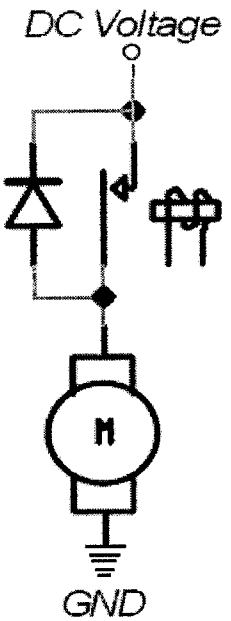
By adding a flyback diode the current has a path to continue flowing through coil until the stored energy is used up. The diode also clamps the voltage across the coil to about 0.7V protecting the electronics. The stored energy dissipates quickly in the diode (E = V*I*t). The current stops flowing and the relay turns off. The diode should be able to handle the coil current for a short time and switch relatively fast. Note: A resistor or zener diode can be placed in series with the diode to use up the stored energy quicker. This increases the amplitude of the voltage spike above 0.7V but the energy is used up quicker (i.e. the voltage spike won't last as long). Usually it doesn't matter if the relay takes 1ms or 100ms to turn off.

The schematic below illustrates how current flows in a relay. Fig A shows the current flow when the transistor is on. Fig B shows the current flow when the transistor is off. Notice how the diode completes the current loop.

Note: An automotive ignition coil uses the stored energy in an inductor combined with a step up transformer to power the spark plugs in your car. By quickly collapsing the magnetic filed over 20,000V can be generated.



Note: If the load is DC and inductive add a fly back diode across the relay contacts (as shown below). The **inductive kickback** from the load will shorten the lifetime of the relay contacts if the diode isn't present.



2.2 CIRCUIT DESCRIPTION

This circuit is designed to control the load. The load may be motor or any other load. The load is turned ON and OFF through relay. The relay ON and OFF is controlled by the pair of switching transistors (BC 547). The relay is connected in the Q2 transistor collector terminal. A Relay is nothing but electromagnetic switching device which consists of three pins. They are Common, Normally close (NC) and Normally open (NO).

The normally open (NO) pin connected to load. When high pulse signal is given to base of the Q1 transistors, the transistor is conducting and shorts the collector and emitter terminal and zero signals is given to base of the Q2 transistor. So the relay is turned OFF state. The circuit diagram for relay is given in the fig 7.2.

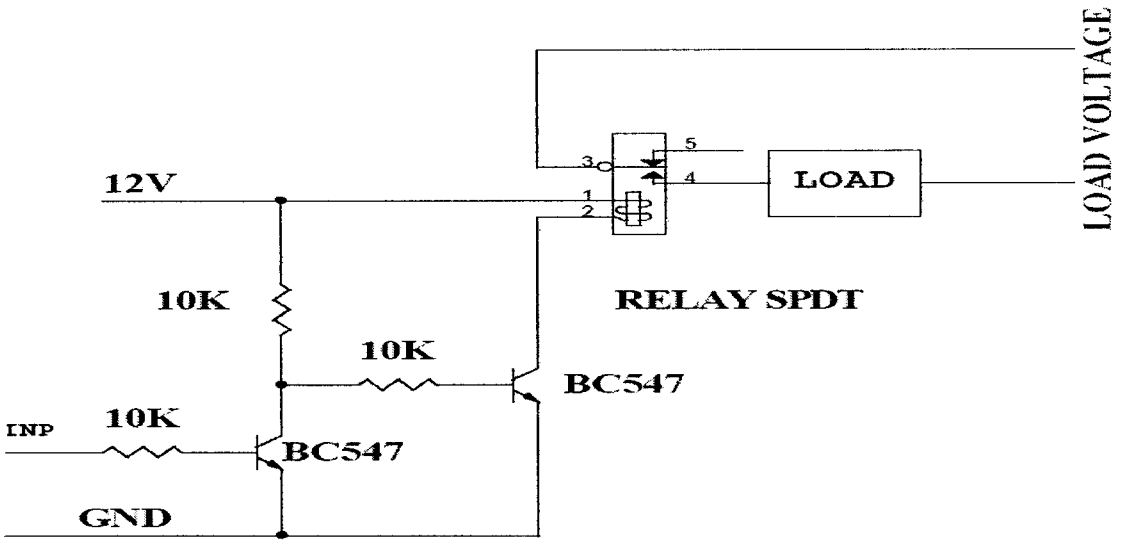


Fig 2.3 CIRCUIT FOR RELAY

2.3 WORKING:

When low pulse is given to base of transistor Q1 transistor, the transistor is turned OFF. Now 12v is given to base of Q2 transistor so the transistor is conducting and relay is turned ON. Hence the common terminal and NO terminal of relay are shorted. Now load gets the supply voltage through relay.

CHAPTER 3

SENSORS

3.1 LM35 - TEMPERATURE SENSOR:

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in deg Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of g/4degC at room temperature and g*/4degC over a full b55 to a150degC temperature range.

Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 mA from its supply, it has very low self-heating, less than 0.1degC in still air.

The LM35 is rated to operate over a b55deg to a150degC temperature range, while the LM35C is rated for a b40deg to a110degC range (b10deg with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-202 package.

3.1.1 CIRCUIT DESCRIPTION:

Features:

- Calibrated directly in Celsius (Centigrade)

- Linear 10.0 mV/degC scale factor

- 0.5degC accuracy guaranteed (at a25degC)

- Rated for full b55deg to a150degC range- Suitable for remote applications

- Low cost due to wafer-level trimming

- Operates from 4 to 30 volts

- Less than 60 mA current drain

- Low self-heating, 0.08degC in still air

- Nonlinearity only (g/4degC typical).

- Low impedance output, 0.1 X for 1 mA load

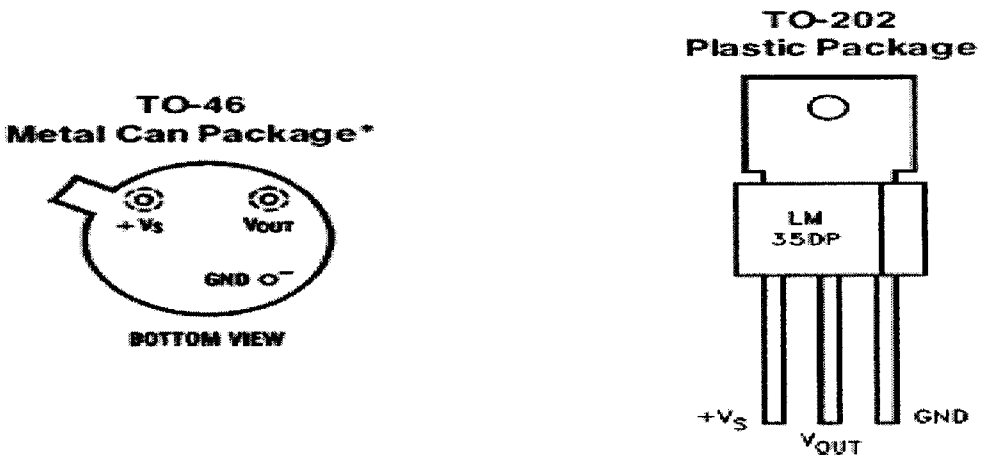


Fig 3.1 LM3\5 Temperature Sensor

3.1.2 WORKING:

Each temperature sensors sense the real time temperatures of the relay terminals. Since there are six terminals to be considered we here use six sensors individually for each terminal. The ambient temperature given as set point is compared with each of the sensor values and if any of the value exceeds the ambient temperature value the trip signal is issued. The values of the six sensors are updated to the monitoring system periodically for future reference.

3.2 MEASUREMENT OF VOLTAGE

The measurement of voltage at any one of the terminals (NC or NO) may be suitably measured using a common voltage sensor. But in our project we apply only 5V across the relay contacts for the process of testing. So, we can directly give the relay contact points to microcontroller for measuring the voltage.

Normally here we take NC as the measuring contact point and we give the signal to microcontroller. The voltage across the NC point varies regularly as 0V and 5V.

If there is a relay failure there will be a constant 0V or constant 5V across the NC contact point. Here at this point we may stop the counter section to estimate the number of cycles the relay had withstood. This also gives the life span of the particular relay.

CHAPTER 4

MICROCONTROLLER

MICROCONTROLLER

4.1 INTRODUCTION

A microcontroller is a complete microprocessor system built on a single IC. Microcontrollers were developed to meet the need for microprocessors to be put into low cost products. Building a complete microprocessor system on a single chip substantially reduces the cost of building simple products, which use the microprocessor's power to implement their function, because the microprocessor is a natural way to implement many products. This means the idea of using a microprocessor for low cost products comes up often. But the typical 8-bit microprocessor based system, such as one using a Z80 and 8085 is expensive. Both 8085 and Z80 system need some additional circuits to make a microprocessor system. Each part carries costs of money. Even though a product design may require only very simple system, the parts needed to make this system as a low cost product.

To solve this problem microprocessor system is implemented with a single chip microcontroller. This could be called microcomputer, as all the major parts are in the IC. Most frequently they are called microcontroller because they are used they are used to perform control functions.

The microcontroller contains full implementation of a standard MICROPROCESSOR, ROM, RAM, I/0, CLOCK, TIMERS, and also SERIAL PORTS. Microcontroller also called "system on a chip" or "single chip microprocessor system" or "computer on a chip".

4.2 PIC MICROCONTROLLER

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

The main advantage of CMOS and RISC combination is low power consumption resulting in a very small chip size with a small pin count. The main advantage of CMOS is that it has immunity to noise than other fabrication techniques.

Various microcontrollers offer different kinds of memories. EEPROM, EPROM, FLASH etc. are some of the memories of which FLASH is the most recently developed. Technology that is used in pic16F877 is flash technology, so that data is retained even when the power is switched off. Easy Programming and Erasing are other features of PIC 16F877.

4.2.1 CORE FEATURES

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle

- Operating speed: DC – 20 MHz clock input

  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,

  Up to 368 x 8 bytes of Data Memory (RAM)

  Up to 256 x 8 bytes of EEPROM data memory
- Pin out compatible to the PIC16C73/74/76/77
- Interrupt capability (up to 14 internal/external
- Eight level deep hardware stack
- Direct, indirect, and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC Oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS EPROM/EEPROM technology
- Fully static design
- In-Circuit Serial Programming (ICSP) via two pins

- Only single 5V source needed for programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.5V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption

4.2.2 PERIPHERAL FEATURES

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules

  Capture is 16-bit, max resolution is 12.5 ns,

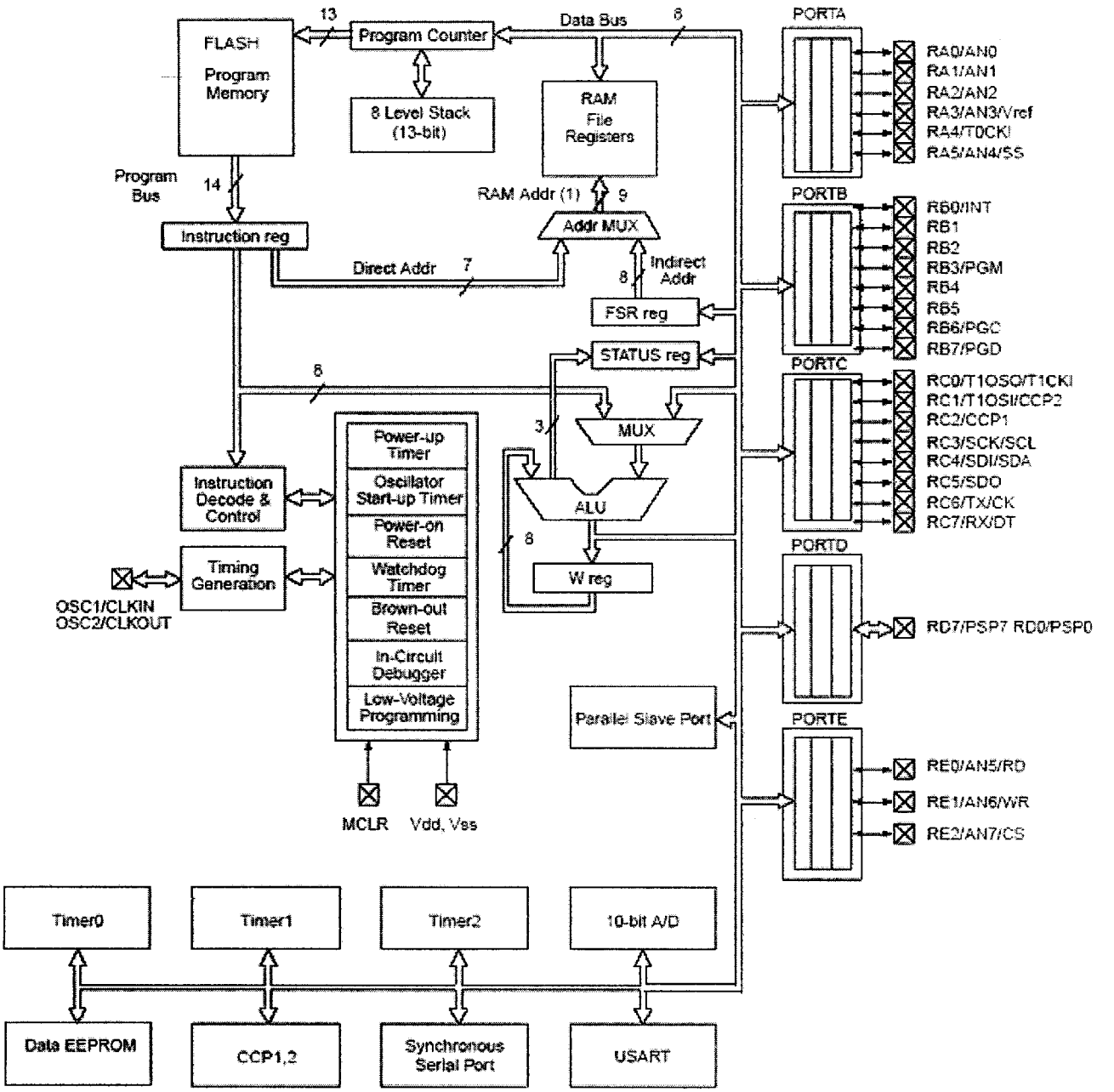
  Compare is 16-bit, max resolution is 200 ns,

  PWM max. Resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter

- Synchronous Serial Port (SSP) with SPI. (Master Mode) and I2C. (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with

  9- Bit addresses detection.
- Brown-out detection circuitry for Brown-out Reset (BOR).

4.3 ARCHITECTURE OF PIC 16F877



Note 1: Higher order bits are from the STATUS register.

FIG 4.1 ARCHITECTURE OF PIC 16F877

4.4 PIN CONFIGURATION

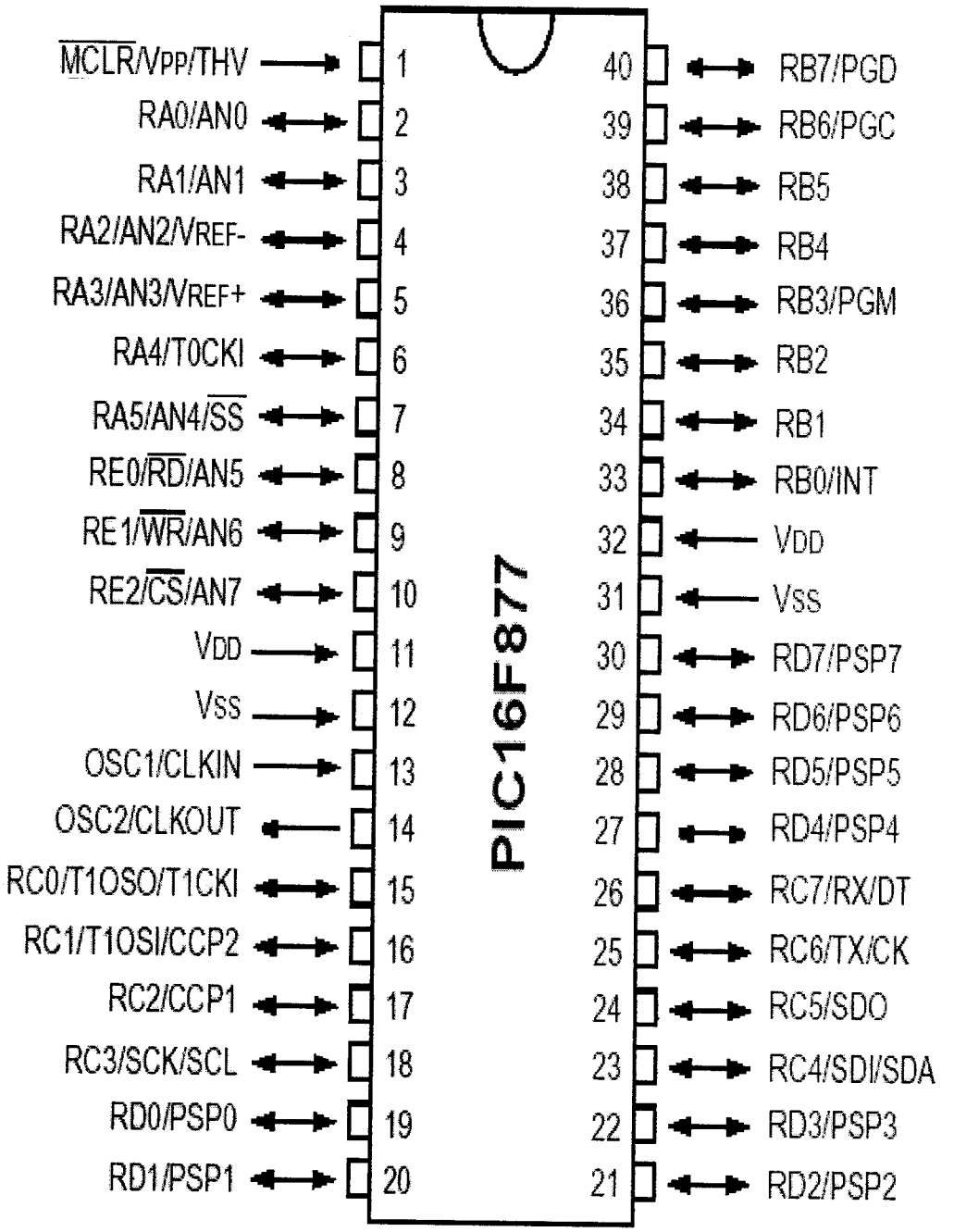


FIG. 4.2 PIN DIAGRAM

4.5 INPUT/OUTPUT PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin. The pins can be selected as input or output port by setting the bits in the respective TRIS registers.

PORTA AND THE TRISA REGISTER

PORTA is a 6-bit wide bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (=1) will make the corresponding PORTA pin an input, i.e., put the corresponding output driver in a Hi-impedance mode. Clearing a TRISA bit (=0) will make the corresponding PORTA pin an output, i.e., put the contents of the output latch on the selected pin.

PORTB AND TRISB REGISTER

PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (=1) will make the corresponding PORTB pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISB bit (=0) will make the corresponding PORTB pin an output, i.e., put the contents of the output latch on the selected pin. Three pins of PORTB are multiplexed with the Low Voltage Programming function; RB3/PGM, RB6/PGC and RB7/PGD. Each of the PORTB pins have a weak internal pull-up. A single control bit can turn on all the pull-ups.

This is performed by clearing bit RBPU. The weak pull-up is automatically turned off when the port pin is configured as an output.

PORTC AND THE TRISC REGISTER

PORTC is an 8-bit wide bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (=1) will make the corresponding PORTC pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISC bit (=0) will make the corresponding PORTC pin an output, i.e., put the contents of the output latch on the selected pin. PORTC is multiplexed with several peripheral functions. PORTC pins have Schmitt Trigger input buffers.

PORTD AND TRISD REGISTERS

This section is not applicable to the 28-pin devices. PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORTD can be configured as an 8-bit wide microprocessor Port (parallel slave port) by setting control bit PSPMODE (TRISE, bit 4). In this mode, the input buffers are TTL.

PORTE AND TRISE REGISTER

PORTE has three pins RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7, which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become control inputs for the microprocessor port when bit PSPMODE (TRISE, bit 4) is set. In this mode, the user must make sure that the TRISE (0 to 2) bits are set (pins are configured as digital inputs). Ensure ADCON1 is configured for digital I/O. In this mode the input buffers are TTL.

PORTE pins are multiplexed with analog inputs. When selected as an analog input, these pins will read as '0's.

4.6 MEMORY ORGANISATION:

There are three memory blocks in each of the PIC16F877 MUC's. The program memory and Data Memory have separate buses so that concurrent access can occur.

PROGRAM MEMORY:

The PIC16f877 devices have a 13-bit program counter capable of addressing 8K *14 words of FLASH program memory. Accessing a location above the physically implemented address will cause a wraparound.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

DATA MEMORY:

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the special functions Registers. Bits RP1 (STATUS, bit 6) and RP0 (STATUS, bit 5) are the bank select bits.

RP1:RP0	Banks
00	0
01	1
10	2
11	3

Each bank extends up to 7Fh (1238 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain special function registers. Some frequently used special function registers from one bank may be mirrored in another bank for code reduction and quicker access.

4.7 WORKING OF THE MICROCONTROLLER:

This is a 40 pin microcontroller. Pin 11 and 32 are given Vcc and pins 12 and 31 are connected with ground and pins 13 and 14 are connected to a crystal oscillator for the functioning of the microcontroller. The command lines to the LCD are connected with port E lines (pins 8 to 10) and the data lines of LCD is connected with port D lines (pins 19 to 21 and 27 to 30).The sensor input is obtained through port A pins (pins 2 to 4). Pin 1 is used to reset the microcontroller. MAX232 is connected through serial pins 25 and 26.

Microcontroller is programmed with embedded C. A compiler is used to convert the high level language into assembly language which is acceptable by microcontroller and program is stored in program memory of the microcontroller. The microcontroller sends the command signals such as clear the display, R/W signal to the LCD display.

The sensor inputs act as interrupt lines. If any one of the interrupt lines is active then the microcontroller identifies which sensor is detected and the corresponding pre-stored information is sent to MAX232 through serial pins.

CHAPTER 5

RS232 COMMUNICATION

5.1 INTRODUCTION

The RS232 is used to interface the micro controller part with the computer for providing the set point and also to monitor the process variables.

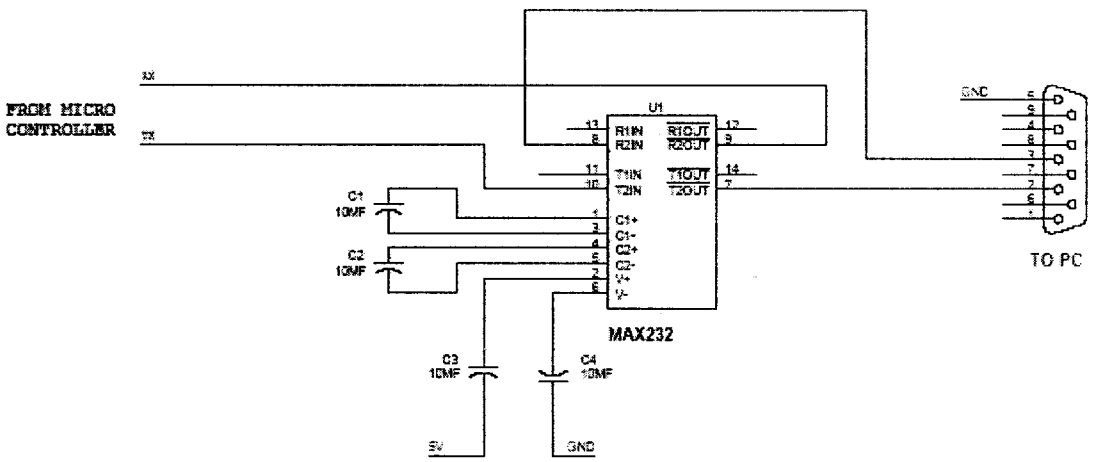


FIG 5.1 RS232 CONNECTION CIRCUIT

RS232

In telecommunications, **RS-232** is a standard for serial binary data interconnection between a DTE (Data terminal equipment) and a *DCE* (Data Circuit-terminating Equipment). It is commonly used in computer serial ports.

5.2 SCOPE OF THE STANDARD

The Electronic Industries Alliance (EIA) standard RS-232-C [3] as of 1969 defines:

- Electrical signal characteristics such as voltage levels, signaling rate, timing and slew-rate of signals, voltage withstand level, short-circuit behavior, maximum stray capacitance and cable length
- Interface mechanical characteristics, pluggable connectors and pin identification
- Standard subsets of interface circuits for selected telecom application

The standard does not define such elements as character encoding (for example, ASCII, Baudot or EBCDIC), or the framing of characters in the data stream (bits per character, start/stop bits, parity). The standard does not define protocols for error detection or algorithms for data compression.

The standard does not define bit rates for transmission, although the standard says it is intended for bit rates lower than 20,000 bits per second. Many modern devices can exceed this speed (38,400 and 57,600 bit/s being common, and 115,200 and 230,400 bit/s making occasional appearances) while still using RS-232 compatible signal levels.

Details of character format and transmission bit rate are controlled by the serial port hardware, often a single integrated circuit called a UART that converts data from parallel to serial form. A typical serial port includes specialized driver and receiver integrated circuits to convert between internal logic levels and RS-232 compatible signal levels.

5.3 CIRCUIT DESCRIPTION

In this circuit the MAX 232 IC used as level logic converter. The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA 232 voltage levels from a single 5v supply. Each receiver converts EIA-232 to 5v TTL/CMOS levels. Each driver converts TLL/CMOS input levels into EIA-232 levels.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

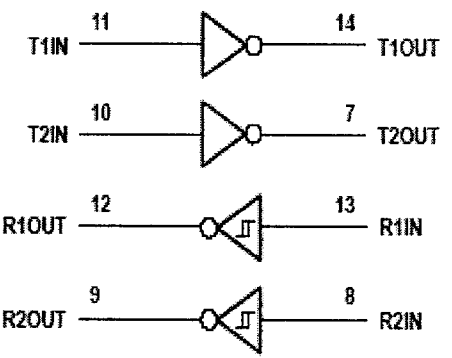
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



In this circuit the microcontroller transmitter pin is connected in the MAX232 T2IN pin which converts input 5v TTL/CMOS level to RS232 level. Then T2OUT pin is connected to reviver pin of 9 pin D type serial connector which is directly connected to PC.

In PC the transmitting data is given to R2IN of MAX232 through transmitting pin of 9 pin D type connector which converts the RS232 level to 5v TTL/CMOS level. The R2OUT pin is connected to receiver pin of the microcontroller. Likewise the data is transmitted and received between the microcontroller and PC or other device vice versa.

5.4 WORKING OF MAX232:

MAX232 is connected to the serial port pins of microcontroller. PC accepts the signal level of +12v. But the output level of microcontroller is 0V to 5V. So an interface called MAX232 is used here to convert 0V to 5V from microcontroller as +12V to PC and vice versa.

Two pairs of capacitors are connected externally to MAX232. One pair of capacitor is used to boost up the signal to PC and to change its polarity as bipolar signal. Another pair of capacitor is used to attenuate the signal and used to change the bipolar signal into TTL signal.

CHAPTER 6

SOFTWARE IMPLEMENTATION

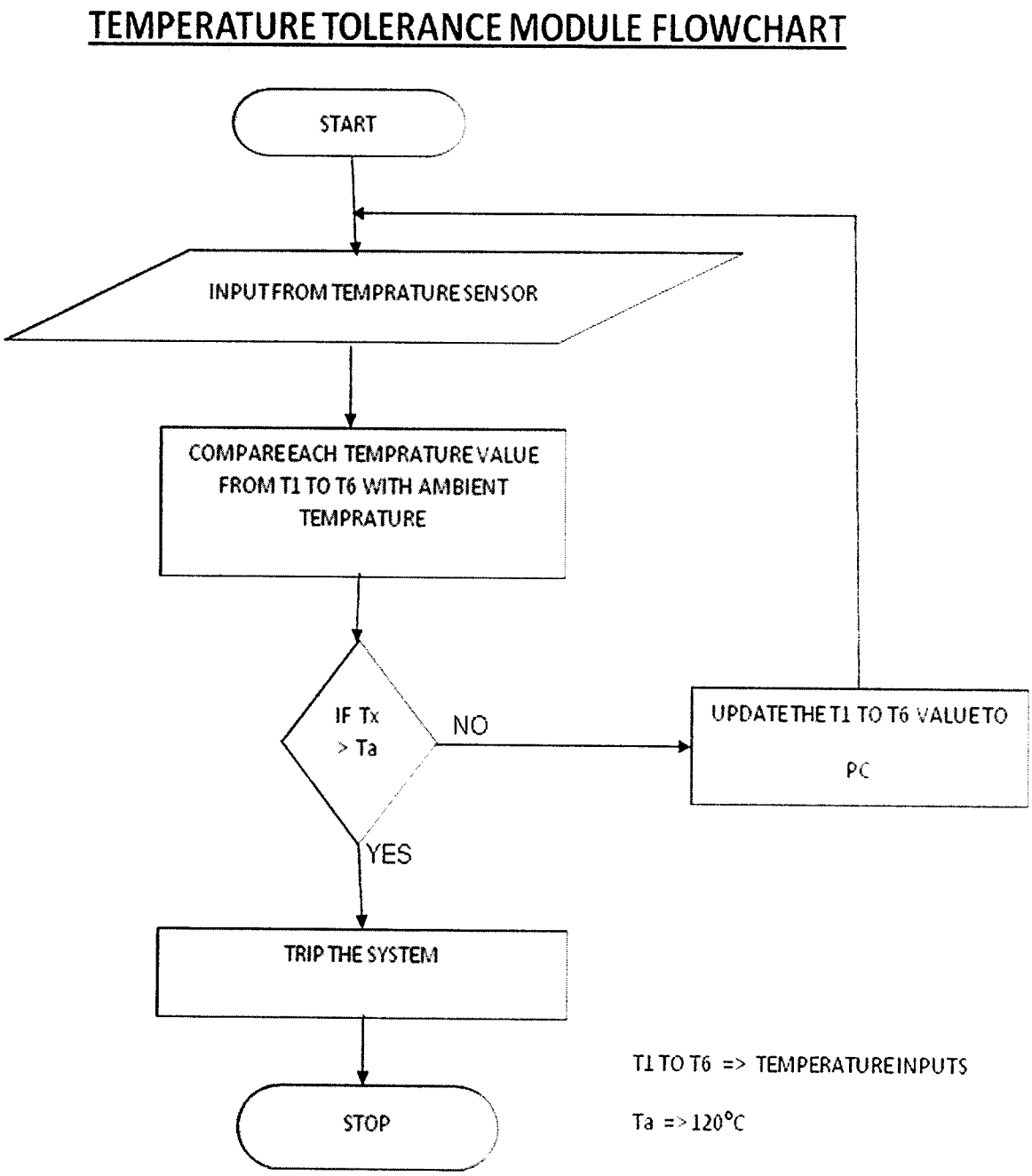
6.1 INTRODUCTION

LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) is a platform and development environment for a visual programming language from National Instruments. Execution is determined by the structure of a graphical block diagram (the LV-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Lab VIEW ties the creation of user interfaces (called front panels) into the development cycle. Lab VIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: a block diagram, a front panel, and a connector panel. Controls and indicators on the front panel allow an operator to input data into or extract data from a running virtual instrument. However, the front panel can also serve as a programmatic interface. Thus a virtual instrument can either be run as a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the given node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

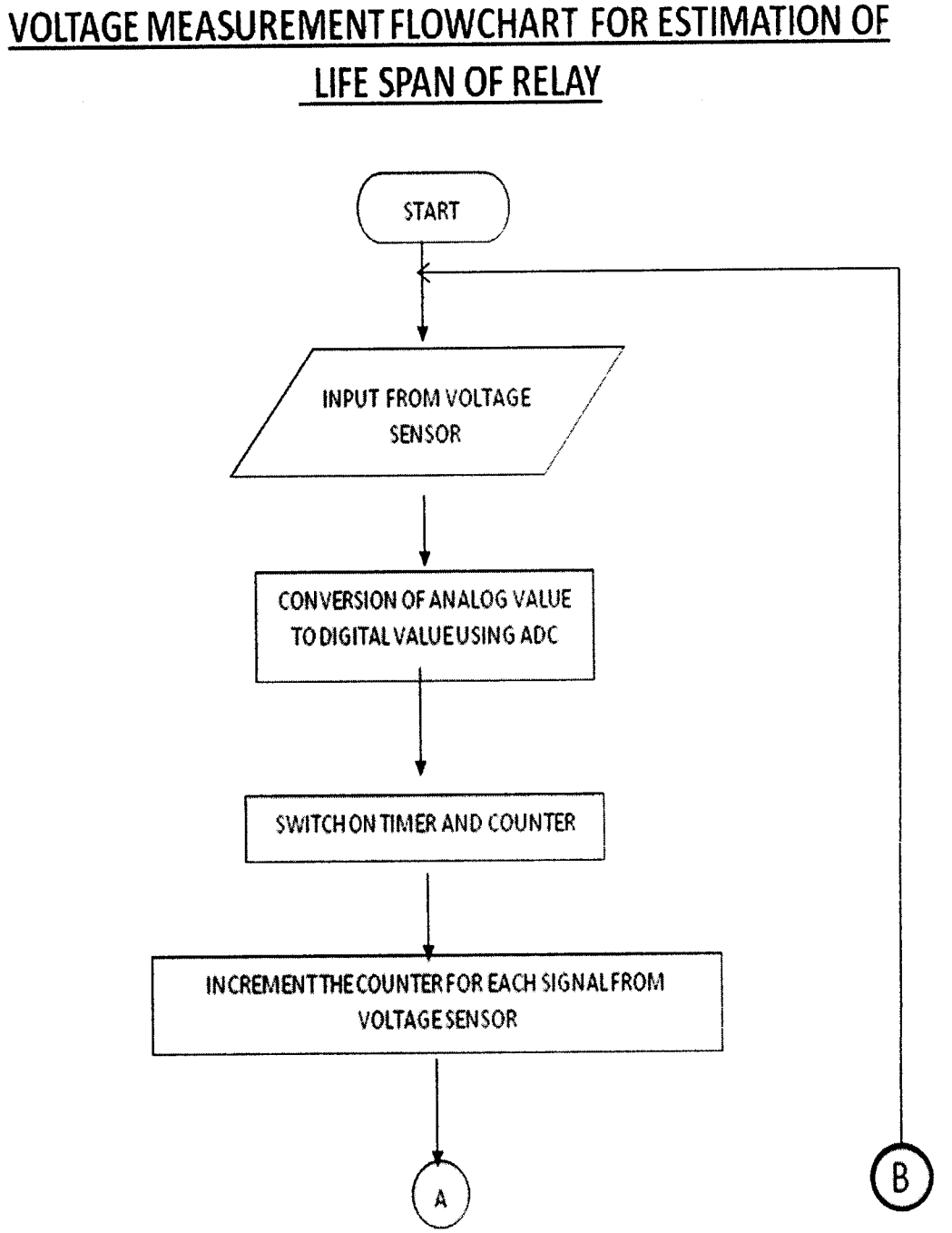
The graphical approach also allows non-programmers to build programs simply by dragging and dropping virtual representations of lab equipment with which they are already familiar.

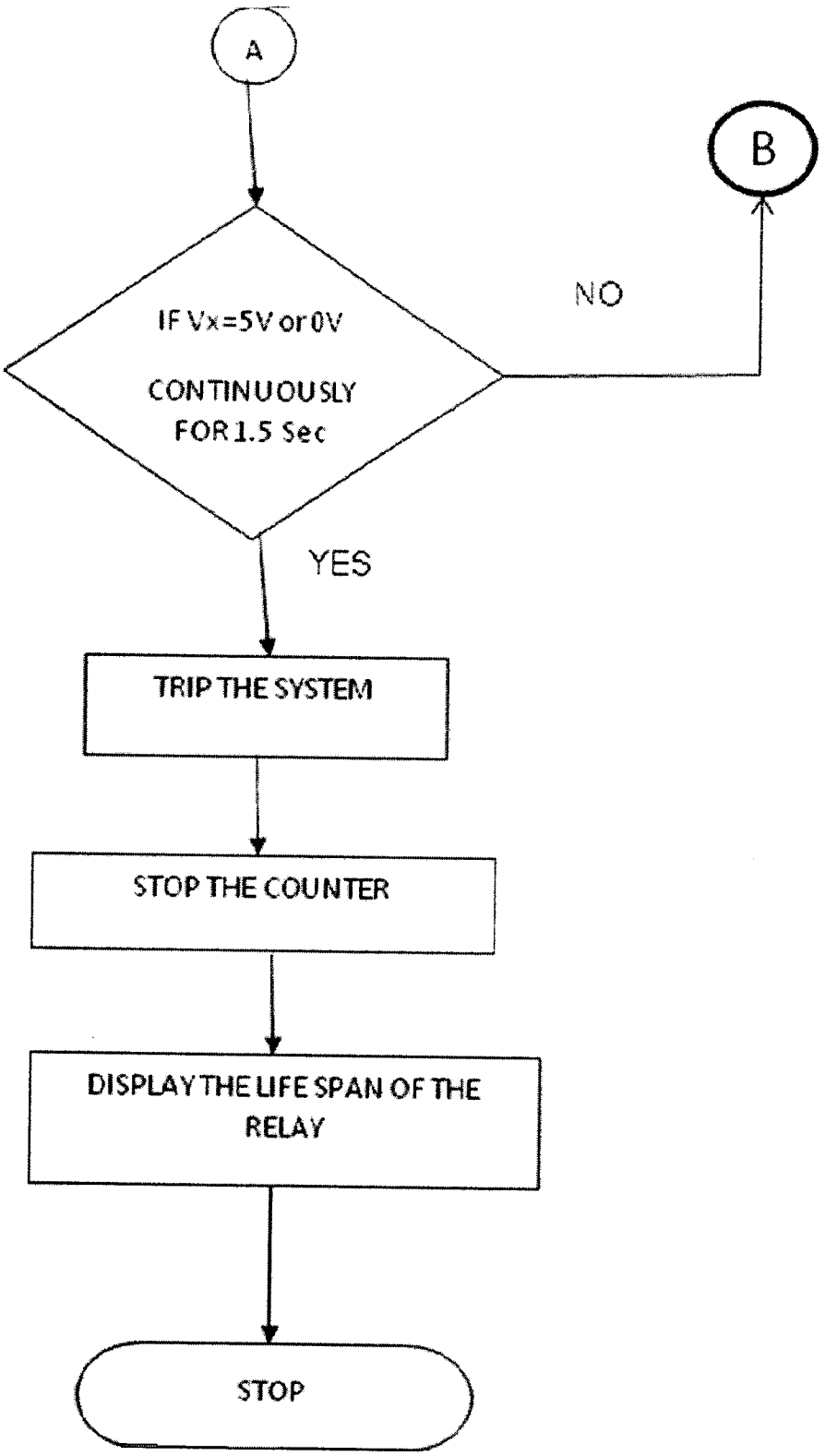
6.2 FLOW CHART:

6.2.1 TEMPERATURE:

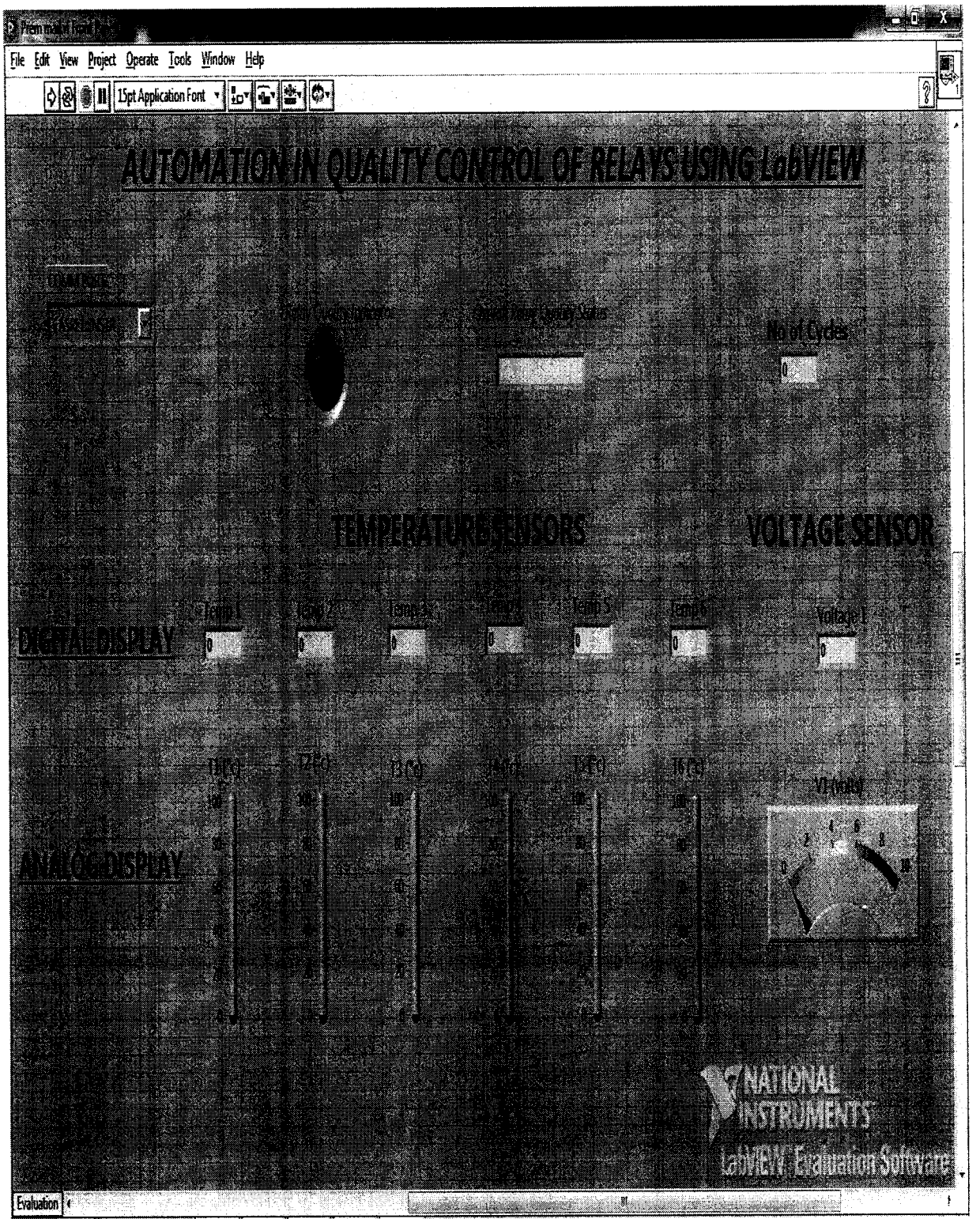


6.2.1 VOLTAGE:

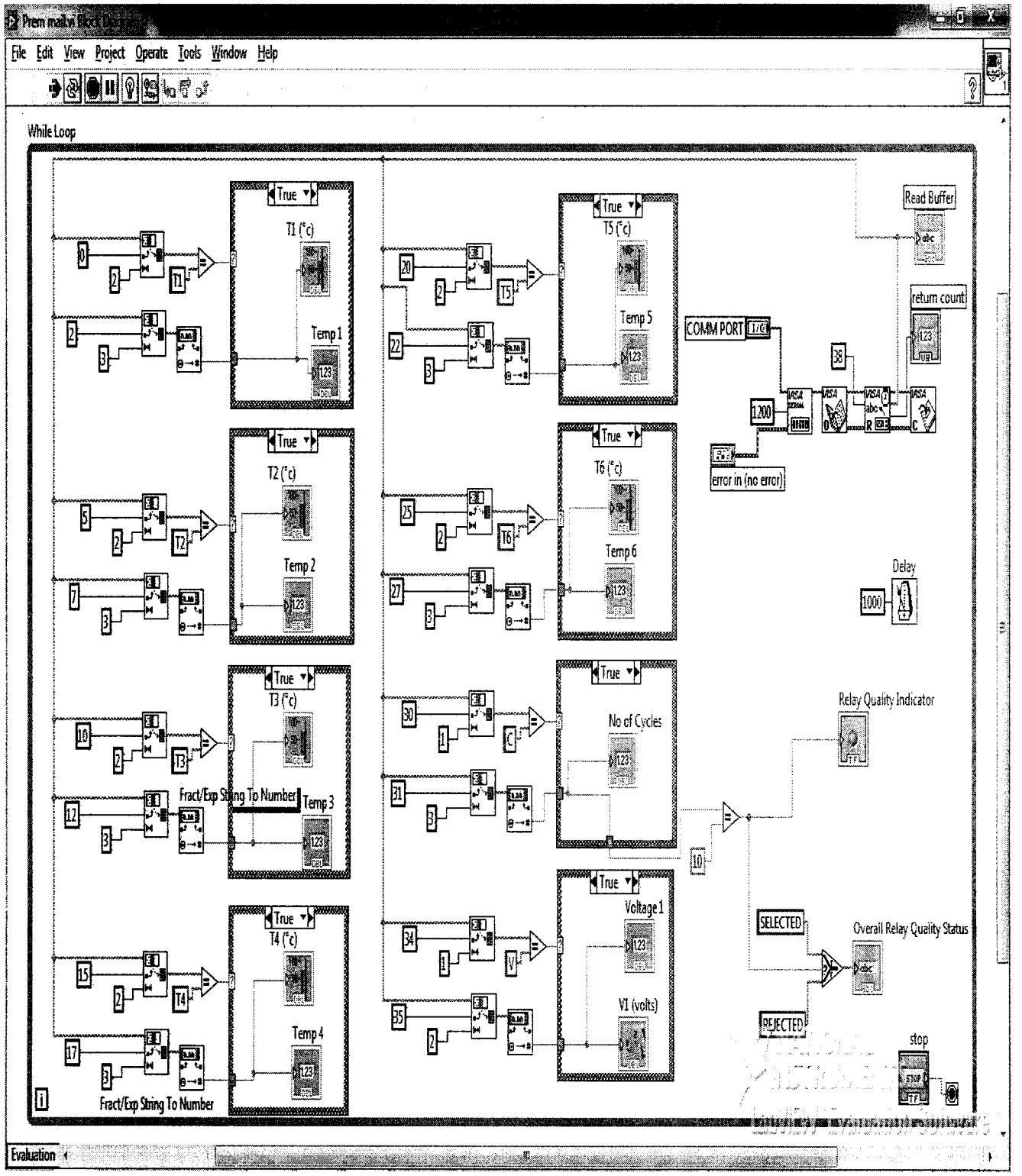

VOLTAGE MEASUREMENT FLOWCHART FOR ESTIMATION OF LIFE SPAN OF RELAY
START
INPUT FROM VOLTAGE SENSOR
CONVERSION OF ANALOG VALUE TO DIGITAL VALUE USING ADC
SWITCH ON TIMER AND COUNTER
INCREMENT THE COUNTER FOR EACH SIGNAL FROM VOLTAGE SENSOR
A
B


A
B
IF Vx=5V or 0V
CONTINUOUSLY
FOR 1.5 Sec
NO
YES
TRIP THE SYSTEM
STOP THE COUNTER
DISPLAY THE LIFE SPAN OF THE
RELAY
STOP

6.3 FRONT PANEL – OVERALL VIEW



6.4 BLOCK DIAGRAM - OVERALL VIEW

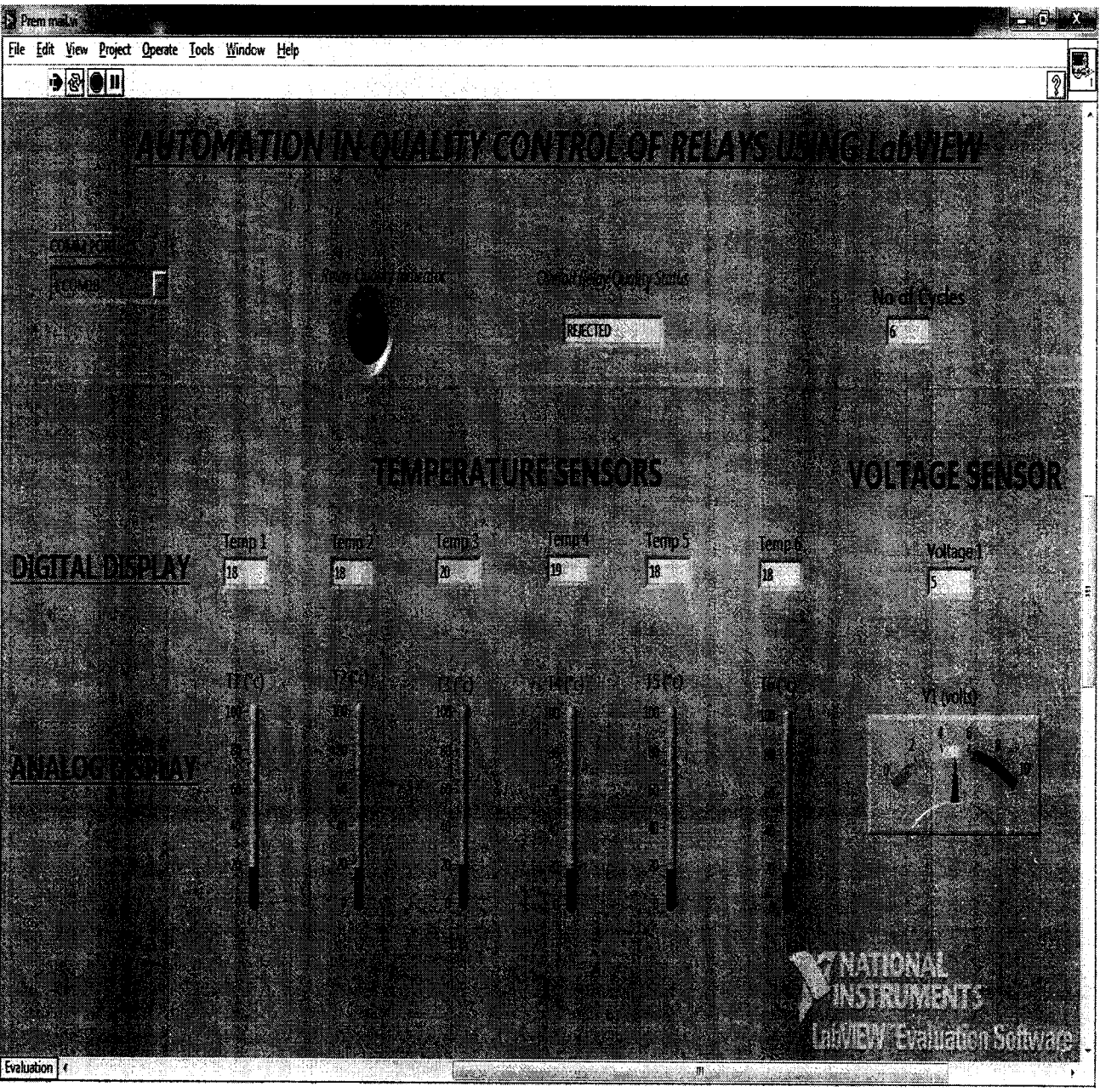


CHAPTER 7

RESULTS AND DISCUSSIONS

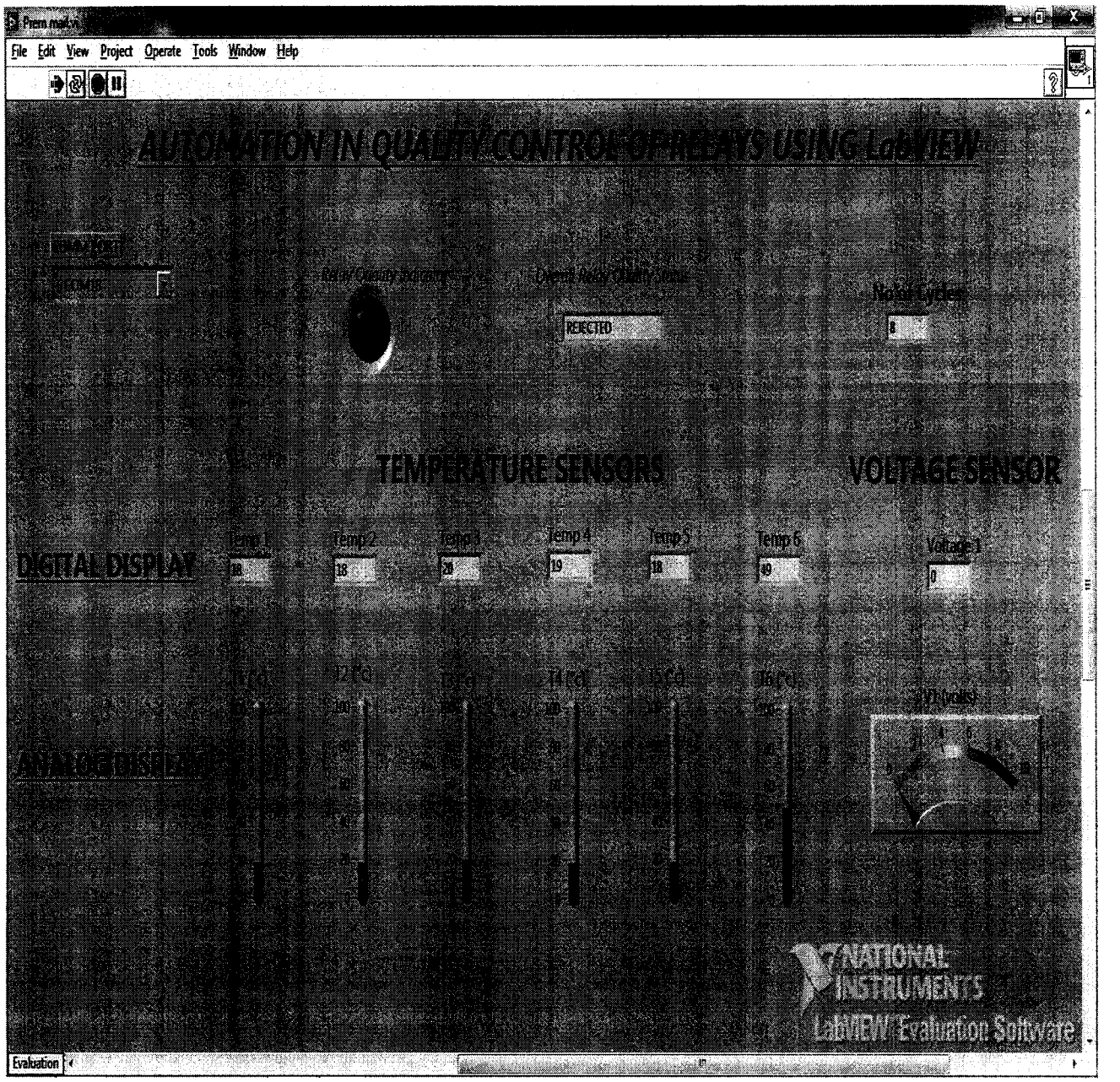
7.1 DISQUALIFIED RELAY - FAILURE DUE TO CONSTANT VOLTAGE:

Here the common of the testing relay get melted and sticks to the NC terminal more than the switching time. So the microcontroller generates a trip signal to cut-off the supply voltage.



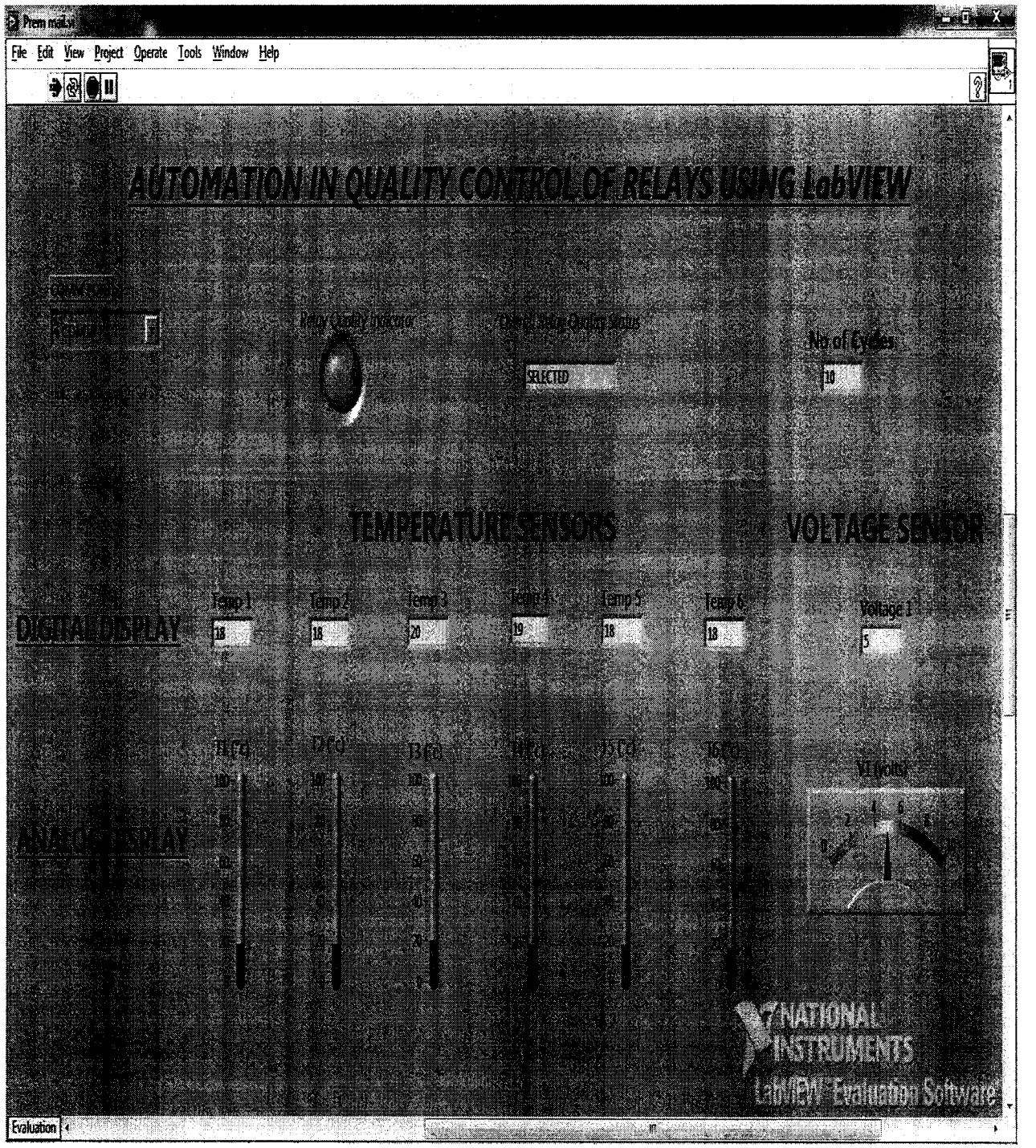
7.2 DISQUALIFIED RELAY - FAILURE DUE TO HIGH TEMPEATURE:

When any one of the six temperature terminal values exceeds the set point temperature (say 120 deg C), the microcontroller issues a trip signal to the test relay.

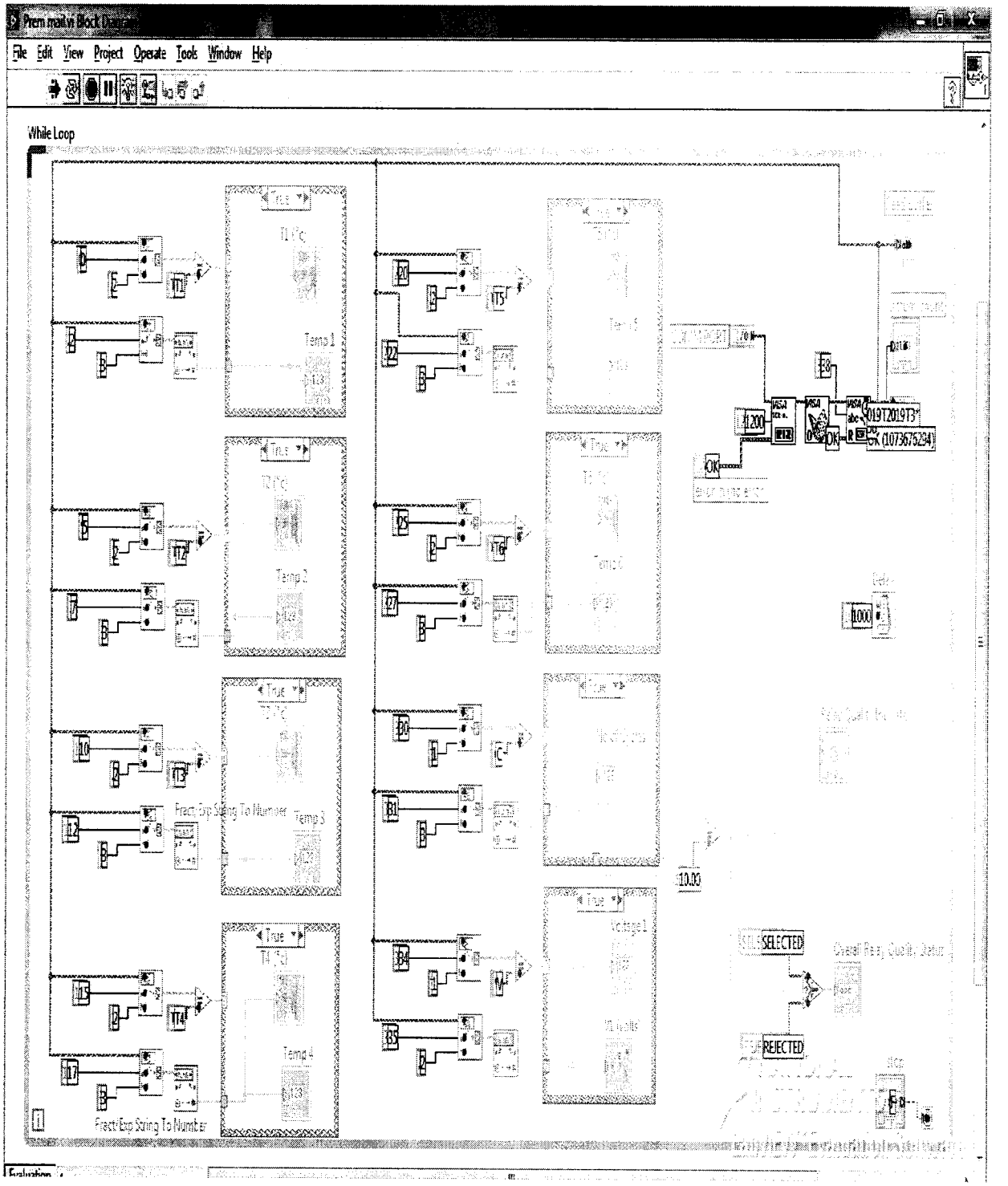


7.3 QUALIFIED RELAY IN QUALITY TEST:

Here the test relay withstands the given number of switching cycles, voltage range and temperature range.



7.4 BLOCK DIAGRAM –EXECUTION:



CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8. CONCLUSION AND FUTURE SCOPE:

- The testing of relays for their temperature tolerance and life span of working were automated and thus it can be implemented into the Relay manufacturing industries for effective quality production of Relays.

- We believe that in future, more parameters will be taken into account like pressure; solidity etc and relays could be manufactured with least of human work.

CHAPTER 9

BIBLOGRAPHY

BIBLIOGRAPHY

IEEE PAPERS USED FOR REFERENCE

- **Automatic testing and lifetime management of protection relays.** By Tout, G.D. Sokhey, I.S. Webb, A.C. (Developments in Power System Protection, 1993., Fifth International Conference)

- **A fully digital real-time simulator for protective relay testing.** By Kuffel, R. Giesbrecht, J. Maguire, T. Wierckx, R.P. Forsyth, P.A. McLaren, P.G. RTDS Technol. Inc., Winnipeg, Man. (Developments in Power System Protection, Sixth International Conference)

- **Advanced simulation tool for relay testing.** By Xiaolei Liu Osman, A.H. Malik, O.P. Univ. of Calgary, Calgary, AB, Canada. (Power Symposium, 2008. NAPS '08. 40th North American)

WEBSITES REFERRED

- http://www.isatest.com
- http://www.relaytestingsolutions.com
- http://www.howstuufworks.com
- http://www.ni.com/labview

BOOKS REFERRED

- **Robert Bishop – Labview 2009** Student. Edition, Prentice Hall, 2010.

CHAPTER 10

APPENDICES

10.1CODING FOR MICROCONTROLLER:

```
#include<pic.h>

//#include<hitech.h>

#define rs     RD6

#define en      RD7

#define datas    PORTB

#define command()    rs=0;

#define data()       rs=1;

#define lcd_strobe()  en=1;delay(500);en=0;

#define RLY1 RD3

/*#define LED2 RC2

#define switch1 RD0

#define switch2 RD1*/

unsigned int a,b,c,count,temper7;bit disp1,disp2,rl,stop,fail,enddisp,vlt;unsigned int
digit1,digit2,digit3,volt;

unsigned char temper1,temper2,temper3,temper4,temper5,temper6;

bank2 const char name1[]={"  TEMPERATURE   "};

bank2 const char name2[]={"    SENSOR      "};

bank2 const char name3[]={"1:  2:  3:    "};

bank2 const char name4[]={"4:  5:  6:    "};

bank2 const char name5[]={"count:         "};

//bank2 const char name6[]={"           "};

bank2 const char name7[]={"TEST OK        "};

bank2 const char name8[]={"TEST IS FAIL    "};

char name9[]={"VOLTAGE:       "};
```

```
void convertASCII(unsigned int);
void delay(unsigned int );
void lcd_write(unsigned char );
void lcd_read(unsigned char );
void lcd_display(unsigned char ,unsigned char );
void lcd_init();
void send(unsigned char );
void adctemperature1();
void adctemperature2();
void adctemperature3();
void adctemperature4();
void adctemperature5();
void adctemperature6();
void adcvolt();
void delay(unsigned int y)
{
  while(y--);
}
void lcd_read(unsigned char y)
{
 datas=y;
 command();
 lcd_strobe();
}
void lcd_write(unsigned char y)
{
```

```
 datas=y;
 data();
 lcd_strobe();
}
void lcd_display(unsigned char a,unsigned char b)
{
 lcd_read(a);
 lcd_write(b);
}
void lcd_init()
{
  lcd_read(0x38);
  lcd_read(0x06);
  lcd_read(0x0c);
  lcd_read(0x01);
}
void send(unsigned char a)
{
TXREG=a;
while(!TRMT);
TRMT=0;
TXIF=0;
}
void adctemperature1()
{
 unsigned char tempr1;
```

```
ADCON0=0X81;
tempr1=0;
ADRESL=0x00;
ADRESH=0x00;
ADGO=1;
while(ADGO);
tempr1=(((256*ADRESH)+ADRESL));
if(tempr1<=482)
 {
temper1=tempr1/3.2;
 }
 if((tempr1>482)&&(tempr1<=493))
 {
temper1=tempr1/2.4;
 }
 if((tempr1>493)&&(tempr1<=502))
 {
temper1=tempr1/1.64;
 }
  if((tempr1>502)&&(tempr1<=510))
 {
temper1=tempr1/1.25;
 }
 if((tempr1>510)&&(tempr1<=518))
 {
temper1=tempr1/1.0;
```

```
}
if((tempr1>518)&&(tempr1<=526))
{
temper1=tempr1/0.86;
}
if((tempr1>526)&&(tempr1<=532))
{
temper1=tempr1/0.75;
}
if((tempr1>532)&&(tempr1<=540))
{
temper1=tempr1/0.66;
}
if((tempr1>540)&&(tempr1<=547))
{
temper1=tempr1/0.6;
}
if((tempr1>547))
{
temper1=tempr1/0.54;
}
}
void adctemperature2()
{
unsigned char tempr2;
ADCON0=0X89;
```

```
tempr2=0;
ADRESL=0x00;
ADRESH=0x00;
ADGO=1;
while(ADGO);
tempr2=(((256*ADRESH)+ADRESL));
if(tempr2<=482)
{
temper2=tempr2/3.2;
}
if((tempr2>482)&&(tempr2<=493))
{
temper2=tempr2/2.4;
}
if((tempr2>493)&&(tempr2<=502))
{
temper2=tempr2/1.64;
}
 if((tempr2>502)&&(tempr2<=510))
{
temper2=tempr2/1.25;
}
if((tempr2>510)&&(tempr2<=518))
{
temper2=tempr2/1.0;
}
```

```
if((tempr2>518)&&(tempr2<=526))
 {
temper2=tempr2/0.86;
 }
 if((tempr2>526)&&(tempr2<=532))
 {
temper2=tempr2/0.75;
 }
  if((tempr2>532)&&(tempr2<=540))
 {
temper2=tempr2/0.66;
 }
 if((tempr2>540)&&(tempr2<=547))
 {
temper2=tempr2/0.6;
 }
  if((tempr2>547))
 {
temper2=tempr2/0.54;
 }
}
void adctemperature3()
{
 unsigned char tempr3;
 ADCON0=0X91;
```

```
ADRESL=0x00;
ADRESH=0x00;
ADGO=1;
while(ADGO);
tempr3=(((256*ADRESH)+ADRESL));
if(tempr3<=482)
{
temper3=tempr3/3.2;
}
if((tempr3>482)&&(tempr3<=493))
{
temper3=tempr3/2.4;
}
if((tempr3>493)&&(tempr3<=502))
{
temper3=tempr3/1.64;
}
 if((tempr3>502)&&(tempr3<=510))
{
temper3=tempr3/1.25;
}
if((tempr3>510)&&(tempr3<=518))
{
temper3=tempr3/1.0;
}
if((tempr3>518)&&(tempr3<=526))
```

```
 {
temper3=tempr3/0.86;
 }
 if((tempr3>526)&&(tempr3<=532))
 {
temper3=tempr3/0.75;
 }
  if((tempr3>532)&&(tempr3<=540))
 {
temper3=tempr3/0.66;
 }
 if((tempr3>540)&&(tempr3<=547))
 {
temper3=tempr3/0.6;
 }
  if((tempr3>547))
 {temper3=tempr3/0.54
 }
 void adctemperature4()
 {
  unsigned char tempr4;
  ADCON0=0X99;
  tempr4=0;
  ADRESL=0x00;
  ADRESH=0x00;
  ADGO=1;
```

```
while(ADGO);
tempr4=(((256*ADRESH)+ADRESL));
 if(tempr4<=482)
 {
temper4=tempr4/3.2;
 }
 if((tempr4>482)&&(tempr4<=493))
 {
temper4=tempr4/2.4;
 }
 if((tempr4>493)&&(tempr4<=502))
 {
temper4=tempr4/1.64;
 }
  if((tempr4>502)&&(tempr4<=510))
 {temper4=tempr4/1.25;
 }
 if((tempr4>510)&&(tempr4<=518))
 {
temper4=tempr4/1.0;
 }
 if((tempr4>518)&&(tempr4<=526))
 {
temper4=tempr4/0.86;
 }
 if((tempr4>526)&&(tempr4<=532))
```

```
 {
temper4=tempr4/0.75;
 }
  if((tempr4>532)&&(tempr4<=540))
 {
temper4=tempr4/0.66;
 }
 if((tempr4>540)&&(tempr4<=547))
 {
temper4=tempr4/0.6;
 }
  if((tempr4>547))
 {
temper4=tempr4/0.54;
 }
 }
 void adctemperature5()
 {
  unsigned char tempr5;
  ADCON0=0XA1;
  tempr5=0;
  ADRESL=0x00;
  ADRESH=0x00;
  ADGO=1;
  while(ADGO);
  tempr5=(((256*ADRESH)+ADRESL));
```

```
 if(tempr5<=482)
 {
temper5=tempr5/3.2;
 }
 if((tempr5>482)&&(tempr5<=493))
 {
temper5=tempr5/2.4;
 }
 if((tempr5>493)&&(tempr5<=502))
 {
temper5=tempr5/1.64;
 }
  if((tempr5>502)&&(tempr5<=510))
 {
temper5=tempr5/1.25;
 }
 if((tempr5>510)&&(tempr5<=518))
 {
temper5=tempr5/1.0;
 }
 if((tempr5>518)&&(tempr5<=526))
 {
temper5=tempr5/0.86;
 }
 if((tempr5>526)&&(tempr5<=532))
 {
```

```
temper5=tempr5/0.75;
 }
  if((tempr5>532)&&(tempr5<=540))
 {
temper5=tempr5/0.66;
 }
 if((tempr5>540)&&(tempr5<=547))
 {
temper5=tempr5/0.6;
 }
  if((tempr5>547))
 {
temper5=tempr5/0.54;
 }
 }
 void adctemperature6()
 {
 unsigned char tempr6;
  ADCON0=0XA9;
  tempr6=0;
  ADRESL=0x00;
  ADRESH=0x00;
  ADGO=1;
  while(ADGO);
  tempr6=(((256*ADRESH)+ADRESL));
   if(tempr6<=482)
```

```
{
temper6=tempr6/3.2;
}
if((tempr6>482)&&(tempr6<=493))
{
temper6=tempr6/2.4;
}
if((tempr6>493)&&(tempr6<=502))
{
temper6=tempr6/1.64;
}
if((tempr6>502)&&(tempr6<=510))
{
temper6=tempr6/1.25;
}
if((tempr6>510)&&(tempr6<=518))
{temper6=tempr6/1.0;
}
if((tempr6>518)&&(tempr6<=526))
{
temper6=tempr6/0.86;
}
if((tempr6>526)&&(tempr6<=532))
{
temper6=tempr6/0.75;
}
```

```
 if((tempr6>532)&&(tempr6<=540))
 {
temper6=tempr6/0.66;
 }
 if((tempr6>540)&&(tempr6<=547))
 {
temper6=tempr6/0.6;
 }
  if((tempr6>547))
 {
temper6=tempr6/0.54;
 }
 }
 void adcvolt()
 {
 unsigned int v;
  ADCON0=0XB1;
  v=0;
  ADRESL=0x00;
  ADRESH=0x00;
  ADGO=1;
  while(ADGO);
  v=(((256*ADRESH)+ADRESL));
  volt=v/20;
  }
 void interrupt isr(void)
```

```
{
//10ms timer1 isr
if(TMR1IF)
{
a++;b++;c++;
if(a>=800)
{
 RLY1=1;rl=1;vlt=1;disp2=1;disp1=0;b=0;//c=0;
}
if(b>=300)
{
disp2=0;disp1=1;
}
if(a>=1000)
{
a=0;
RLY1=0;vlt=0;
}
/*if(b<=300)
{
disp1=1;disp2=0;
}
if(b>300)
{
disp2=1;disp1=0;
}*/
```

```
//if(b>=600)
//{
//b=0;
//}
TMR1L = 0xEF; //10ms
TMR1H = 0xD8;
TMR1IF=0;}
}
void convertASCII(unsigned int conv)
{
digit1=conv%1000/100+0x30;
digit2=conv%100/10+0x30;
digit3=conv%10+0x30;
send(digit1);
send(digit2);
send(digit3);
}
void main()
{
unsigned char i,j,n;
unsigned int de;
//unsigned int
digit1,digit2,digit3/*,digit11,digit12,digit13,digit21,digit22,digit23,digit31,digit32,digit33,digit41,digit42,
digit43,
//digit51,digit52,digit53,digit61,digit62,digit63*/;
TRISA=0xFF;
TRISB=0X00;
```

```
TRISD=0x03;
TRISC=0x80;
TRISE=0x01;
 PORTA=0x00;
PORTB=0X00;
PORTD=0X00;
PORTE=0x00;
PORTC=0x00;
 ADCON1=0X80;
RCSTA=0x80;
TXSTA=0x20;
BRGH=1;
SPBRG=207;
CREN=0;
TXEN=1;
PEIE=1;
GIE=1;
//INTE=1;
T1CON = 0x05;
PIR1 &= 0xFE;
PIE1 |= 0x01;
TMR1L = 0xEF; //10ms
TMR1H = 0xD8;
fail=0;stop=0;
lcd_init();
for(i=0;i<=16;i++)
```

```
{
lcd_display((0x80+i),name1[i]);
}
for(i=0;i<=16;i++)
{
lcd_display((0xc0+i),name2[i]);
}
lcd_read(0x01);
for(i=0;i<=16;i++)
{
lcd_display((0x80+i),name3[i]);
}
for(i=0;i<=16;i++)
{
lcd_display((0xC0+i),name4[i]);
}
while(1)
   {
 adcvolt();
if((RLY1==1)&&(c>=900))
{
c=0;
if(volt==0)
{
fail=1;
//lcd_read(0x01);
```

```
}
}
if((RLY1==0)&&(c>=150))
{
if(volt>0)
{
fail=1;c=0;
//lcd_read(0x01);
}}
if(disp1==1)
{
lcd_read(0x01);
for(i=0;i<=16;i++)
{
lcd_display((0x80+i),name3[i]);
}
for(i=0;i<=16;i++)
{
lcd_display((0xC0+i),name4[i]);
}
while(disp1)
{
adctemperature1();
adctemperature2();
adctemperature3();
adctemperature4();
```

```
adctemperature5();
adctemperature6();
if((temper1>=45)||(temper2>=45)||(temper3>=45)||(temper4>=45)||(temper5>=45)||(temper6>=45))
{
fail=1;
//lcd_read(0x01);
}
if(fail==1)
{
lcd_read(0x01);
while(fail)
{
for(i=0;i<=16;i++)
{
lcd_display((0x80+i),name8[i]);
}
RLY1=0;
GIE=0;
}
}
 //lcd_display(0x82,temper1%10000/1000+0x30);
 //lcd_display(0x83,temper1%1000/100+0x30);
lcd_display(0x82,temper1%100/10+0x30);
lcd_display(0x83,temper1%10+0x30);
//lcd_display(0x8B,temper2%10000/1000+0x30);
//lcd_display(0x8C,temper2%1000/100+0x30);
```

```
lcd_display(0x87,temper2%100/10+0x30);
lcd_display(0x88,temper2%10+0x30);
//lcd_display(0xc2,temper3%10000/1000+0x30);
//lcd_display(0xc3,temper3%1000/100+0x30);
lcd_display(0x8c,temper3%100/10+0x30);
lcd_display(0x8D,temper3%10+0x30);
//lcd_display(0xcB,temper4%10000/1000+0x30);
 //lcd_display(0xc2,temper4%1000/100+0x30);
lcd_display(0xc2,temper4%100/10+0x30);
lcd_display(0xc3,temper4%10+0x30);
//lcd_display(0xcC,temper5%1000/100+0x30);
lcd_display(0xc7,temper5%100/10+0x30);
lcd_display(0xc8,temper5%10+0x30);
//lcd_display(0xcC,temper5%1000/100+0x30);
lcd_display(0xcc,temper6%100/10+0x30);
lcd_display(0xcd,temper6%10+0x30);
send('T');
send('1');
convertASCII(temper1);
send('T');
send('2');
convertASCII(temper2);
send('T');
send('3');
convertASCII(temper3);
send('T');
```

```
send('4');
convertASCII(temper4);
send('T');
send('5');
convertASCII(temper5);
send('T');
send('6');
convertASCII(temper6);
send('C');
convertASCII(temper7);
send('V');
convertASCII(volt);
enddisp=0;
}
}
if(disp2==1)
{
lcd_read(0x01);
temper7=0;
for(i=0;i<=16;i++)
{
lcd_display((0x80+i),name5[i]);
}
for(i=0;i<=16;i++)
{
lcd_display((0xC0+i),name9[i]);
```

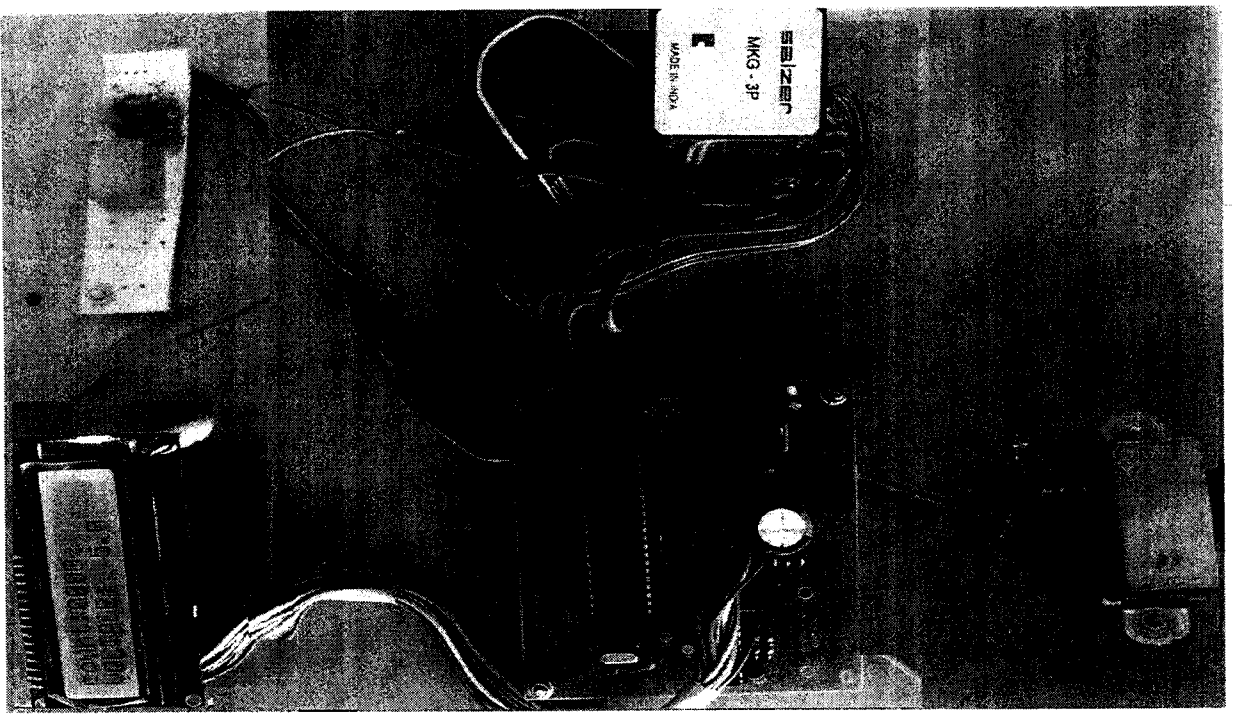
```
}
while(disp2)
{
adcvolt();
temper7=count;
lcd_display(0x86,temper7%10000/1000+0x30);
 lcd_display(0x87,temper7%1000/100+0x30);
lcd_display(0x88,temper7%100/10+0x30);
lcd_display(0x89,temper7%10+0x30);
//lcd_display(0xc9,volt%10000/1000+0x30);
 lcd_display(0xca,volt%100/10+0x30);
lcd_display(0xcb,'.');
 lcd_display(0xcc,volt%10+0x30);
if((RLY1==1)&&(c>=900))
 {
c=0;
if(volt==0)
 {
fail=1;
//lcd_read(0x01);
 }
 }
if((RLY1==0)&&(c>=150))
 {
if(volt>0)
 {
```

```
fail=1;c=0;
//lcd_read(0x01);
}
}
//enddisp=1;
send('T');
send('1');
convertASCII(temper1);
send('T');
send('2');
convertASCII(temper2);
send('T');
send('3');
convertASCII(temper3);
send('T');
send('4');
convertASCII(temper4);
send('T');
send('5');
convertASCII(temper5);
send('T');
send('6');
convertASCII(temper6);
send('C');
convertASCII(temper7);
send('V');
```

```
convertASCII(volt);
enddisp=0;
}
if(rl==1)
{
rl=0;
count++;
}
if(count>10)
{
count=0;stop=1;
lcd_read(0x01);
}
while(stop)
{
for(i=0;i<=16;i++)
{
lcd_display((0x80+i),name7[i]);
}
RLY1=0;
GIE=0;
}
}
}
```

10.2 PHOTOGRAPHS

RELAY QUALITY TESTING SYSTEM



MICRONTROLLER CONNECTION DIAGRAM