1999

Design and development of fuzzy expert system for Handy Board

Aditya Kumar Singh
West Virginia University

Design and Development of Fuzzy Expert System for Handy Board

Aditya Kumar Singh

Thesis

Submitted to the
College of Engineering and Mineral Resources
of West Virginia University
in partial fulfillment of the Requirements for
the degree of

Master of Science in Engineering

Dr. B. Gopalakrishnan, Ph.D., Chair, IMSE
Dr. Ralph W Plummer , Ph.D., IMSE
Dr. Charles F Stanley, Ph.D., MAE

Department of Industrial and Management Systems Engineering
West Virginia University
Morgantown, WV
1999

ABSTRACT

Design and development of Fuzzy Expert System for Handy Board

Aditya Kumar Singh

The recent trend in global manufacturing scenario has resulted in the emergence of many innovative techniques that have revolutionized the manufacturing industry as a whole. A lot of states of art advances have been associated with the increasing use of microprocessors in advance manufacturing systems. Out of many the one, which has predominantly captured the real time, behavioral use is the Fuzzy Logic.

Fuzzy logic implementation in real time environment involves the integration of multi-disciplinary area like computer engineering, mechanical and electronic systems tailored towards smart systems that enable a very precise control on the process. The fuzzy logic based project involves the programming of microprocessor for the control of sequence of operation depending upon the change in the values of environment variables.

In this research, a computer based system for operating a Motorola Handyboard was designed and developed. A user friendly system in a Visual Basic environment is developed which generates a C++ code to be compiled by the microprocessor using the IC compiler of the Handyboard. The program is then downloaded and with the change in the environment variables the change in output can be observed. Visual Basic is used to develop the front end and integrate the C++ code for performing various functionalities. The system has features to compile and download the code to the Handyboard. The system also helps the user to navigate through the software.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Definition

With the advent of information technology in 1980's, microprocessors with enhanced capabilities were introduced into more sophisticated products like NC machines, industrial robots etc improved their efficiency and performance. Today's commercial products are embedded with mechatronic and fuzzy features. In last few years fuzzy logic has captured attention of nearly all the commercial product manufacturers. A typical mechatronic application picks up the signals, processes them and generates forces and motion as outputs.

Fuzzy logic was given the real meaning by Dr. Lotfi Zadeh in the year 1965 in his paper titled [1] "Fuzzy sets, Information and Control Vol. 8." He viewed it as advanced form of multi valued logic. According to him fuzzy logic deals with more approximate reasoning rather than precise modes of the same, making it a decision making tool over a range of transitional values instead of precise ones.

In today's world of advanced cross technology development which is now controlled by consumer driven society- principle's of Fuzzy logic plays a pioneer role. More and more systems are made adaptive to knowledge based system in which deduction to answer the complex problems requires the inference machinery of fuzzy logic.

According to Gupta [2] it can be stated as a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth which can be stated in other words as truth value between completely true and completely false.

1.2 Evolution of Fuzzy Logic

During the past several years, fuzzy logic has found numerous applications in the fields ranging from finance to earthquake engineering. But the striking feature and probably the most important and visible one in today's world is in a realm not anticipated when fuzzy logic idea was conceived. The field of fuzzy logic based process control has been in limelight in various fields.

The basic idea underlying fuzzy logic control was suggested in notes published in 1968 [3] and 1972 [4] and then further detailed in 1973 [5]. The initial implementation of fuzzy logic concept was pioneered by Madaami and Assilan [6] in connection with regulation of steam engine.

As stated by Zadeh in [5] Fuzzy Logic is turning out to be the preferred method because conventional control systems are based on mathematical differential equations which sometimes do not facilitate the translation of human problem solving technique into a computer algorithm. As per Jager fuzzy logic when used with control systems uses linguistic approach which allows us to express the desired control actions in words.

1.3 Advantages of Fuzzy Logic

Using fuzzy logic, system designer can realize lower development cost, superior features and better end product performances. Most important feature is that product can be brought to market faster and more cost effectively. Several factors because of which fuzzy logic has become so desirable to the system designer are as follows:

1. Describe the model solution to problems without having to use (using) complex mathematical models for systems and development.

2. Optimize the known solution in order to obtain a simpler and more effective implementation.
3. Simplify the system design process, thereby decreasing development costs.
4. Make the system more descriptive so that the system is more convenient to manage, maintain, and upgrade, and easier to differentiate with less risk.
5. Have a high fault tolerance and a better trade off between systems robustness and system sensitivity.
6. Provide products with powerful features and performance within a price range unmatched by other solution.

Fuzzy Logic provides a method to construct algorithms in a user friendly way and provides the ability to capture the non–linear control behavior of humans which has proven to be appropriate for many complex tasks. Having a design method for controllers which is closer to human thinking and perception can reduce development time and requires less skilled personnel to design controllers. The economical benefit of this is trivial. It can be further inferenced that robustness of human controllers is primarily due to their ability to adapt to changing environment and their learning capability.

1.4 Applications of Fuzzy Logic

In today's world fuzzy logic has found application in every realm of life, from washing machines to control of delicate processes, in navigation and avionics. In Japan in particular, the use of fuzzy logic in control processes is being pursued in many applications areas among them are automatic train operation (Hitachi), vehicle control (Sugeno laboratory at Tokyo Institute of Technology), robot control (Hirola Laboratory at Hosei

University), speech recognition (Ricoh), Universal Controller (Fuji) and stabilization control (Yamakawa lab at Kumamtono University) [9].

Some excellent examples of implementations of fuzzy logic have been Sendai subway system. Although, it was challenged that fuzzy system cannot be used in safety driven situation the system has been in operation since 1986 and is more accurate and has doubled the comfort index and reduced power consumption by 10%.

Recently NASA has flown experimental payloads with fuzzy logic based temperature control devices. Around the world all major automotive manufacturers are implementing fuzzy concepts for cruise control, engine spark advances, engine idling and active shock absorption.

In recent months researchers at Ohio State University have taken steps towards developing a system using fuzzy logic to help pilots regain control of aircraft following a major system malfunction. They are working on the basis that fuzzy logic can be used to give inexact instructions to machines which can then decide the level of performance of machines that is appropriate to the level of failure that has occurred.

1.5 Basics of Mechatronics

The word "Mechatronics" was first coined by Japanese in 1970 to describe a new technology fusion. In the beginning mechatronics was viewed as combination of mechanics and electronics. Mechatronic is synergetic combination of precision mechanical engineering, electronic control and system thinking in the design of products and processes.

In 1970’s mechatronics as stated by Steven [37] was concerned mostly with servo technology used in products such as automatic door openers, vending machines and auto focus cameras.

In 1980’s, as information technology was introduced engineers begin to embed microprocessor’s in mechanical systems to improve there performance. Numerically controlled machines and robots became more compact while automotive applications such as electronic engine controls and anti lock breaking systems became wide spread.

By the 1990’s communication technology was added to the mix, yielding products that could be connected in large networks this development made functions such as the remote operations of robotic manipulator arms possible. At the same time, new, smaller even micro scale sensors and actuators technologies are being used increasingly in new products. Micro electro mechanical systems such as tiny silicon accelerometers that trigger automatic air bags, are examples of the latest use.

1.6 Benefits of Mechatronics

The advantages of mechatronics can be best highlighted by the success of the Brintons of Kinderminster, makers of world famous Axminster and Wilton carpets [36]. Brintons, all most uniquely designed and manufactured the carpet making machines themselves. In order to compete in terms of quality and cost it is necessary to drive these machines as fast as possible while retaining precision. It was only by the adoption of the mechatronics that the latest Brinton machinery were able to stake comfortably ahead of their competitors without any compromise to the quality of the product. Some of the advantages as listed by them are as follows:

- simplified mechanical design,
- rapid machine setup,
- commonality of components,
- cost effectiveness,
- rapid development trials,
- optimize performance.

Mechatronic products exhibit certain distinguishing features like the replacement of mechanical functions with electrical ones [37]. This results in lots of advantages as highlighted below:

- greater flexibility,
- ease of redesign or reprogramming,
- ability to implement distributed control in complex systems,
- ability to conduct automated data collection and reporting,
- shorter development cycles,
- improved functionality and performance.

1.7 Features of Handy Board®

The Handy Board version® 1.2 is a single board computer optimized for controlling DC, stepper and servo motors and receiving data from a host of digital and analog sensors. Its miniature size (4.25 by 3.15) low power operation and programmability makes it an ideal choice for control of small and simple intents. It communicates with desk top computer over a standard serial cable, making it suitable for PC based control the prominent features of the Handy board® are as follows:

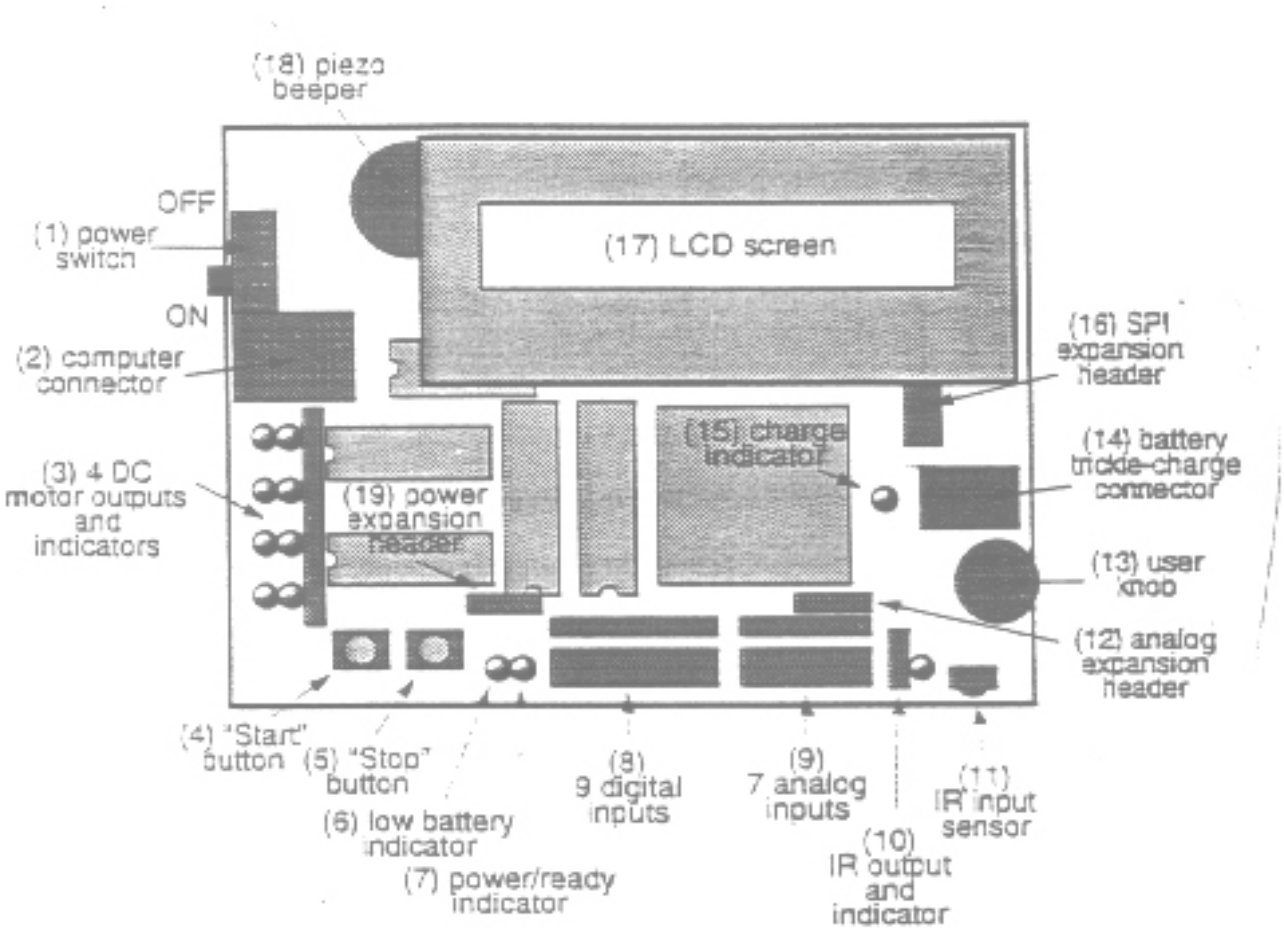

Fig 1.1 Features of Handy Board®

- Input for seven analog sensors and nine digital sensors
- Hardware 40KHz oscillators and drive circuits for IR output and on board receiver.
- Sockets for industry standards and 14pin LCD screen.
- Two user programmable buttons, one knob and piezo beeper.
- 32 kbytes of battery packed CMOS static RAM.
- SPI expansion header and additional power expansion headers.
- Control of four DC motors, at voltages 6 – 36 and up to 600mA of current using software based pulse width modulation technique, motors may be controlled at 100 levels of power from off to on in either direction.

- A RS232 serial port for communicating desk top computers.

1.8 Interactive C

Interactive C (IC) is a C language consisting of a compiler and a run time machine language module . IC has been developed by Randy Sargent of Newton Labs And Fred Martin of MIT [10] unlike assembly language, where programs are first coded, assembled, downloaded and then tested, IC incorporates a command line interface to test interactively various function calls and command lines programs. Interactive C implements a subset of C and works by compiling a pseudo code for a custom stack machine, rather than compiling directly into a native code for a particular processor. This pseudo code is then interpreted by a run time machine language program. This unusual approach to compiler design allows the following features:

- *Interpreted execution* that allows the run time error checking and prevents crashing.
- Small object (machine) code tends to be smaller than a native code representation.
- Multitasking as the pseudo code is stack based, it is easy to programme multiple task all running parallel at the same time.

1.8.1 Multitasking in IC

One of the most prominent features of multitasking is its IC facilities. This means that the processor would be able to virtually run several programs at the same time. While this may not seem possible for a single processor to do this, it is accomplished through time slicing. As the processor works extremely fast it appears to do several things simultaneously by spending a little bit of time on one process then spending another little

bit of time on next process and so on. In this manner each process will get some processor time because each process gets a little “slice”of time, it is known as time slicing. Processes are identified by numbers and can be created and destroyed dynamically at run time. It is also possible for some processes to take over the microprocessor for a long time or defer the execution before the fixed time is elapsed.

1.9 Need for Fuzzy Expert System

Fuzzy rule based system allow a more natural expression of concepts by experts and users then crisp systems as stated by [38]. Fuzzy sets are used to model the vagueness and imprecision present in natural language, thus they can be employed to represent concepts such as *rarely and often.* Fuzzy base systems are converted into expert system by using a collection of fuzzy membership functions and rules instead of boolean logic to reason about the data. Mechatronic on other hand focuses on synthesis of mechanics and electronics. The design and development of smart products acquires a synergistic integration of technologies of all engineering disciplines. Intelligence and flexibility are two essential features in smart electronic products like cameras, cam corders and washing machines. With the advent of technological break through there is a constant pressure for development of user friendly smart products. Fuzzy logic is now more used than ever before in development of controllers for satisfying human expectation and behavioral requirements. In a fuzzy controller these adjustments are handled by a fuzzy rule based expert system, a logical model of thinking processes that a person might go through in the course of manipulating the system. This shift in the focus, from the process to the persons involved, changes the entire approach to the automatic control problems. Combining multi

valued logic, probability theory and knowledge based fuzzy logic control simulates human thinking by incorporating the imprecision inherent in all physical systems.

1.10 Need for Research

Fuzzy logic though originated in 1960's has gained real momentum for providing solutions to real life solutions. Whether it is to deal with objects in real world, or for controlling and modeling real time events as well as interfacing real time events binary concepts values are no longer feasible. The building of real time system requires use of sensors, software's, electronic applications and control engineering. In recent years fuzzy logic has proven well it's broad potential in industrial and automation application. In these application areas engineers rely on proven concepts of discrete event control, they mostly use ladder logic (a programming language resembling electrical wiring schemes). However, these controller work fine when processes are under control and stable conditions. They are not able to cope in other rapid transitional environment since most of the real time phenomenon do not account for discrete values. Here is when fuzzy logic comes into picture with it's attributes of real life like very low, low, medium, high and transitional states between them can be easily captured. The case of nuclear power plants exactly replicates the real time environment where controls have to be monitored for even a fractional rise in conditions of variable parameters for example temperature, pressure and density.

For the implementation of fuzzy logic in specialized fields like nuclear power plants there is a need for development of small fuzzy engine using microprocessors including

Handyboard® from Fred Martin. Also, with Handy board® the different fuzzy design intent could be easily tested and exhibited without a need for big complicated systems.

The system to be developed for present research is envisioned to help user in translating the fuzzy intent into control engineering application in real world environment.

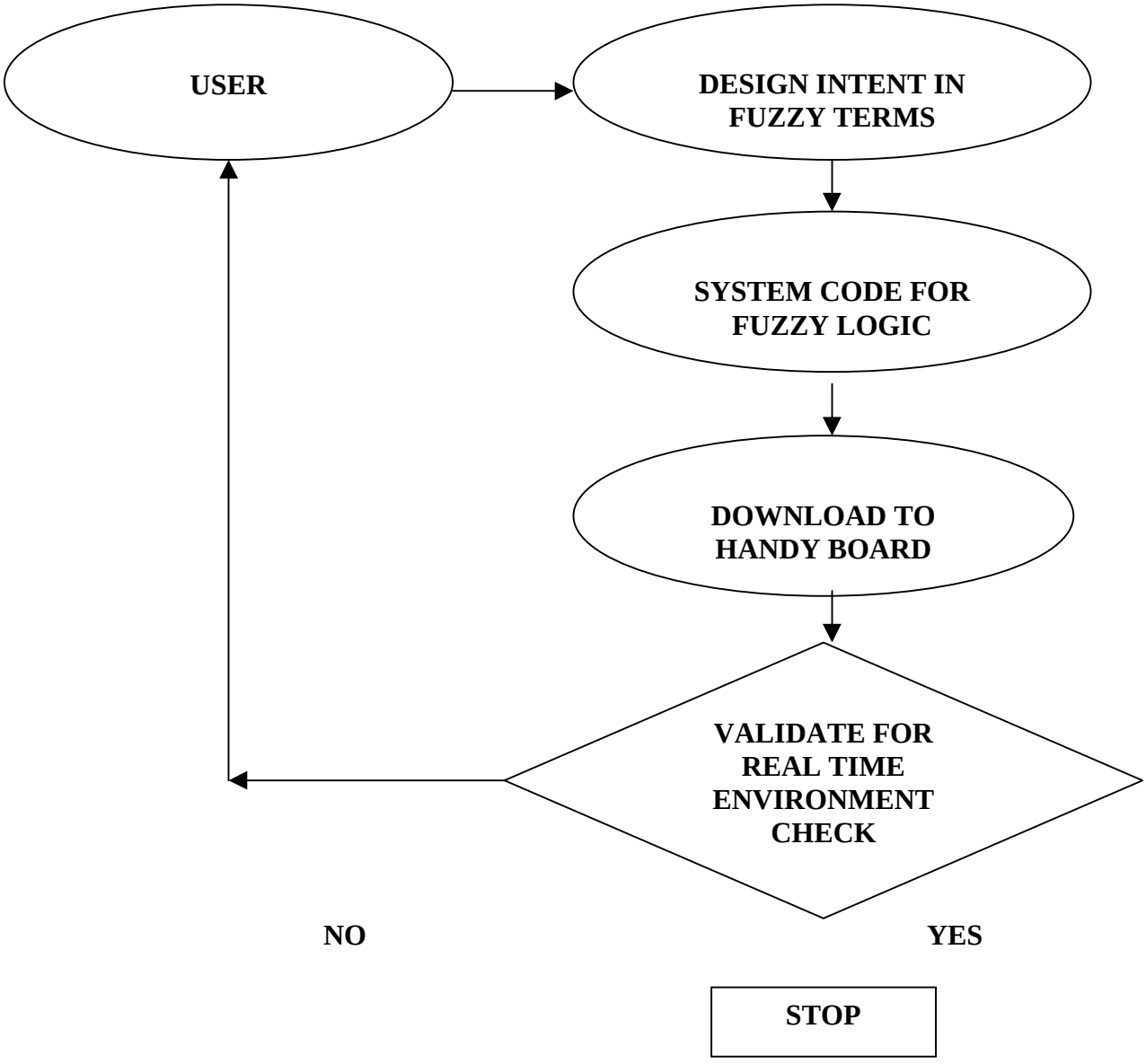

Fig 1.2 System Diagram

1.11 Research Objectives

The main objective of this research is to develop and implement a user-friendly.

The main objective of this research is to develop and implement a user-friendly computer based system using fuzzy logic concepts and to develop the system code, which would further facilitate the effect of changes in associated functionality. This could be expressed as:

1. To develop fuzzy expert system interface to the Handy board®.
2. To design and develop a design intent translator using fuzzy logic principles.
3. Validation of the system to satisfy the user's requirement.

1.12 Conclusions

The major idea behind this research is to design and develop a fuzzy logic based computerized system for testing the design intent of the users using Handy Board® microprocessor. The major focus would be to rapidly prototype the design intent and test for it's functionality.

CHAPTER 2

LITERATURE REVIEW

Today, a lot of interest of industries in fuzzy systems has been generated. More and more companies are making headway to incorporate fuzzy logic in their products. One of the most important reasons for this orientation towards fuzzy control is because competing companies (mostly Japanese) are using or starting to use fuzzy control in competing products.

As stated by Mamdami [12] "*Fuzzy Logic is successful because it replaces the classical PID controller when tuned to required parameters, the parameters of a PID controller affect the shape of the entire control surface. Since the fuzzy logic controller is a rule based controller, the shape of the control surface can be individually manipulated for the different regions of the state space, thus limiting possible effects to neighboring regions only.*"

Fuzzy logic refers to an inapproximate reasoning of systems state in real world environment. In current scenario fuzzy logic systems have turned out to be profitable tool for controlling systems which require human advent expertise to run them as stated by Zadeh [4]. He further states that these systems are basically build on three fundamental features.

1. Linguistic variables are used in additional to numerical variables.
2. Characterization of simple relations between variables by fuzzy conditional statements.
3. Characterization of complex relation by fuzzy algorithms.

2.1 Basics of Fuzzy Logic

2.1.1 Fuzzy Sets

Zadeh [1] introduced fuzzy sets, although others had already recognized the

Zadeh [1] introduced fuzzy sets, although others had already recognized the idea underlying or ideas close to it, mainly by philosophers. Classical set theory which defines membership $\mu_A(x)$ of x of classical set Λ, as subset of the universe X, is defined by:

$$\mu_A(x) = \{ 1 , \text{iff } x \in \Lambda = \{ 0 , \text{iff } x \in \Lambda \quad \text{..... Eq no (1)}$$

This means that an element *x* is either a member of set Λ or not. Classical sets can be also referred as crisp sets. For many classifications it is not clear whether *x* belongs to set Λ or not. A fuzzy set introduced by Zadeh [1] is a set with graded membership in the real interval; $\mu_A(x) \in [0,1]$. A fuzzy set Λ, is a fuzzy subset if X is denoted by

$$\Lambda = \Sigma \mu_A(x_i) / x_I \quad \text{..... Eq.no (2)}$$

Where $\mu_A(x)$ is known as the membership function , and where X is known as universe of discourse. When X is not finite, a fuzzy set Λ is defined by

$$\Lambda = \int_x \mu_A(x) / x \text{.....Eq.no (3)}$$

2.1.2 Operations on Fuzzy Sets

As the operations are defined on classical sets, similar operations are defined on fuzzy sets. The intersections and union of two sets and complement of a set are known from classical set theory. Set theoretic operations like intersection, union and complement are uniquely defined for classical sets and are shown in table below

A	B	A∪B	A∩B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Fig 1.3 Set Theoretic Operation in Classical Theory.

These operations are also defined in fuzzy set theory. However due to the fact that membership values are no longer restricted to (0,1) but can have any value in the interval (0,1), these operators cannot be uniquely defined.

2.1.3 Union and Intersection

The extension of the intersection and union of two classical sets to the intersection and union of two fuzzy sets is not uniquely defined. It is clear that union and intersection operations for fuzzy sets should subject to the intersection and union of classical sets, because a classical set can be seen as special case of a fuzzy set. Zadeh [1] proposed the following definition:

$$\mu_{A \cap B} = \min (\mu_A (x) , \mu_B (x) \text{ intersection} \quad \text{.....Eq no (4)}$$

$$\mu_{A \cup B} = \max(\mu_A (x) , \mu_B (x) \text{ union} \quad \text{..... .. Eq no (5)}$$

2.1.4 Fuzzy Rules

In order to reason with fuzzy logic, fuzzy rules have to be represented by an

application. Such a fuzzy implication has the same truth value as the truth table of the has classical implication in classical logic, but in fuzzy logic these types of statements are often referred to as fuzzy –(*if then)* statements or fuzzy rules. Thus, the fuzzy rules are in the form of an *if–then* statement where the premise and the consequent consist of propositions. The premise contain a combination of proposition by means of logical connectives *and* or *or*.

If x1 is A1 and x_2 is A2 then y is B

When fuzzy sets A1, A2 and B are identified by the membership functions $\mu_{A1}(x1)$ and $\mu_{A2}(x2)$ and $\mu_{B}(y)$, the following fuzzy relation R representing the fuzzy rule can be constructed as follows.

$$R = I\,(T\,(A1, A2)\,B) \qquad \text{Eq no(6)}$$

Where T is based on a general T-norm and I is a general fuzzy implication function.

2.1.5 Fuzzification of Inputs

For any fuzzy rule to be implemented it has to be first fuzzified, implying that the numerical inputs are converted to fuzzy inputs and then a translation from fuzzified output to numerical output. The first translation is known as fuzzification and second is known as defuzzification. The fuzzification of the fuzzy input is the construction of fuzzy relation and compositional rule of inference [7].

2.1.6 Defuzzification of Outputs

Defuzzification is required to translate the fuzzy output of a fuzzy system to a numerical representation to be used for controlling the output of a fuzzy behavior of the

control system. A number of defuzzification methods [7] are available for defuzzifying the output, which include

1. Center of Gravity Defuzzification.
2. Mean of Maxima Defuzzification.
3. Indexed Defuzzification Method.
4. Center of Area Defuzzification.

2.1.6.1 Center of Gravity Defuzzification Method

This defuzzification method works on the principle of calculating centre of gravity of a mass. The difference is that the (point masses are replaced by the membership values. The center of gravity defuzzification method is defined by

$$Cog(B') = \int_y \mu_{B'}(Y)\, Y\, dy \,/\, \int_y \mu_{B'}(Y)\, dy \qquad \text{Eq no (7)}$$

2.1.6.2 Mean of Maxima Method

In this method, which is usually defined only for discrete case, the defuzzified value,$d_{mm}(c)$, is the average of all the values in the crisp set M defined by following equation .

$$d_{(mm)}(C) = \Sigma\ Z\ /\ |M| \qquad \text{Eq no (8)}$$

In the continuos case, when M is given by $d_{(mm)}(C)$ may be defined as the arithmetic average of mean values of all intervals contained in M, including intervals of length of zero. Alternatively, $d_{(mm)}(C)$ may be defined as a weighted average of means values of the intervals, in which the weights are interpreted as the relative lengths of the intervals.

2.1.6.3 Indexed defuzzification methods

This method is particularly used to discriminate part of a fuzzy output of which membership values are below a certain threshold values. This defuzzification method [7] is

only applied on parts of fuzzy output, which have a membership value greater than or equal to $\beta_{t.}$

$$Idfz\ (B, \beta_t) = dfz\ (B' \cap \alpha\text{ - cut }(B', \beta_t), \text{ with } \propto = \beta_t$$

$$= dfz\ (B' \cap \beta_t\text{ - cut }(B'))\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ Eq no (9)}$$

2.1.6.4 Centre of Area defuzzification methods

This is the best defuzzification method available for defuzzifying one-dimensional fuzzy set. In this method the center of area method [7] is defined as

$$\int \mu_{B'}(Y)\, dy = \int y\, \mu_{B'}(Y)\, dy \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ Eq no (10)}$$

Resulting in a numerical value y_{coa} (B) which divides the membership function into two equal parts. It is primarily meant for one-dimensional fuzzy set.

2.1.7 Common Inference Methods

There are number of inference methods [7] available for drawing inferences. The common methods available for drawing inferences are as follows: -

1. Max-min method.
2. Max prods method.

2.1.7.1 Max - min methods

There are basically two operations involved for inferencing - they are aggregation and implication. The compositional rule of inference for max - min method can be written as:

$$\mu_{B'}(Y) \max \min (\beta_k, \mu_{Bk}(Y)) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ Eq no (11)}$$

2.1.7.2 Max- Prod Method

The max product inference method is another commonly applied inference method in fuzzy control. The max-product method is also known as max-dot method. This method is characterized by scaling (product) the consequent B_K of a fuzzy rule r_k degree of fulfillment β_k of that rule and aggregating these result B_k to obtain the fuzzy controller output by means of a max operator:

$$\mu_{B'}(Y) = \max \beta_k * \mu_B(Y) \quad \text{..... Eq no (12)}$$

where * represents the multiplication. The aggregation part is done in same fashion as max-min method.

2.1.8 Fuzzy Propositions

An important concept in fuzzy logic is fuzzy proposition. Fuzzy proposition represent statements like " *x is big"*, where " big " is linguistic label, defined by fuzzy set on universe of discourse of variable x. Fuzzy linguistic labels are also referred [7] to as fuzzy constants, fuzzy terms or fuzzy notions. Fuzzy propositions connect variables with linguistic labels defined for those variables.

These fuzzy logic propositions are the basis for fuzzy logic reasoning. Fuzzy propositions can be combined by means of logical connectives like *and* and *or*. Linguistic modifiers can be used to modify the meaning of the linguistic label used in fuzzy proposition. For example, the linguistic modifier very can be used to change *"x is big" to "x is very big"*.

2.2 Fuzzy Logic Systems

In the last decade the number and scope of application on fuzzy logic controls have increased explosively due to inherent capacity to formalize control algorithm which can tolerate imprecision and uncertainty emulating the cognitive process that human being use everyday.

As stated by Gupta [13] fuzzy systems are in fact, suitable for approximate reasoning for which it is difficult, if not possible to derive an accurate mathematical model. In such cases fuzzy logic is a powerful tool that allows qualitatively expressed control rules to be represented quite naturally, often on basis of simple linguistics descriptions.

In addition when applied to appropriate problems, fuzzy systems have a faster and smoother response than conventional systems. In fact control rules are often simple and do not require great computational complexity. Fuzzy systems are also spreading in several applications areas such as telecommunications networks. Commercial and research trends are emerging, that support adding intelligence to a wide variety of systems (commercial products, industrial controllers and automotive electronics etc.) Major application of fuzzy logic has extended to a large number of control features of industrial automation. They can extend from anti-sway control of cranes to wind energy converter machines. Recently, a 64-ton crane that transports concrete modules for bridges and tunnels over a distance of 500 yards has been automated with fuzzy PLC in Germany [14]. The benefit was capacity gain of about 20% due to faster transportation and an increase in safety. The crane was commissioned in spring 1995 and the crane operator has continuously enabled the fuzzy logic anti-sway controller. This fact is of special importance as not only with technological

feasibility but also psychological aspects are important for the success of an industrial automation solution.

The control of cars and other automotive units have become increasingly more complex due to raised emission standards and constant striving for higher fuel efficiency. As reported by NOK and Nissan [15] the benefits of using fuzzy logic are enormous. The basic idea of fuzzy logic systems is that they first identify the operational condition of the engine by the linguistic variable situation, which contains a number of linguistic terms. Another major application has been in the field of automatic transmission where major automobile players have employed fuzzy logic based control techniques. In 1991, Nissan introduced fuzzy logic controlled automatic five speed transmission systems [16,17]. Honda introduced the concept [18], General Motors first introduced the automatic transmission in 1993 with its Saturn model.

Active stability control systems in cars have also been subject of fuzzy logic systems. Anti-lock braking systems and anti-skid steering systems have made a major breakthrough as reported by the research project sponsored by German car manufacturer.[19]. A 600 rules fuzzy logic controller made the things easier for them.

Peugeot Citroen Corporation of France developed a fuzzy logic system for an intelligent cruise control. The system uses three fuzzy logic blocks with four inputs, one output and 30 rules each. As sensors, the car uses a speed sensor and a single-beam telemeter for the distance from the next car. The actuators using fuzzy logic controller can handle the cruise control under all the tested conditions. [19].

In industrial automation also fuzzy logic technologies enable the efficient and transparent implementation of human control expertise. The individual control loops of

single process variable mostly remain controlled by conventional models such as PIDs. The fuzzy logic system then gives the set of values for these controllers based on the process control expertise substituted in fuzzy logic rules.

Recently software companies have started using Fuzzy logic for knowledge base creation and decision support as stated in [21]. The new simplified approach to Fuzzy database creation uses a " Concept Map" of ideas and connects them with Fuzzy " Rules of Thumb" to automate decision making. Fuzzy logic has recently entered into a field for support of sales and marketing as stated in [22]. A number of sales and marketing executives can now perform there own queries, without having to work through analysts in there IT department. Used in combination with data mining, decision support systems and expert systems, Fuzzy queries can provide a more useful and user friendly computing environment. Companies have also started using Fuzzy logic in non traditional machining methods like electrical discharge machining tools as stated in [23]. Fuzzy logic gives best optimization in EDM's over digital controllers and they were available to improve machining speed upto 200%.

2.3 Fuzzy Logic Applications in Industrial Automation

The subway system of the Japanese City Sendai uses fuzzy logic for train operation. As reported by Yasunobu, P. [9] in comparison with manual control, the fuzzy logic systems are energy efficient. In comparison with conventional control solution, the fuzzy logic controller reaches the stopping points with much higher accuracy.

Goldstar Corporation of Korea uses a fuzzy logic controller [24] for start up and shut down control in a fossil fuel .As stated they are not only able to control the continuous

operation, but the optimal point in time for certain operation is also determined by a fuzzy logic system.

Mitsubishi Heavy Industries Corporation of Japan uses a combination fuzzy logic and conventional controller to control the boron concentration of a nuclear pressurized water reactor. The extension of the conventional controller with fuzzy logic was necessary because the conventional controller could not cope up with the long dead times of the reactors reported by [25]

As stated by Carl Schenck Corporation of Germany uses fuzzy logic system to control a differential distribution weighting means. Bulk material is discharged from buffer by a feeder. The amount of material fed at a certain time is determined by differential of the buffer weight and is used as the actual value for controlling the rotational speed of the feeder. The resulting fuzzy logic based system implements higher precision and a more robust operation compared to the previous conventional solution.

The City of Shanghai in China [26] uses the combination of a fuzzy logic controller and a neural net to optimize the operation of their sewage pump stations. The fuzzy logic system controls the number of active pumps and their power in a most energy efficient way. The neural net predicts the load of the sewage system from the known parameters (day, time, season, rain amount, and other data).

Nippon Steel Corporation of Japan has used fuzzy logic control in a six stand tandem cold strip mill since 1989. The objective is to produce steel with constant thickness. The conventional method requires input of variables that are difficult to measure and that are therefore taken as assumption values only. The implemented fuzzy logic controller showed more constant steel thickness compared to the manually operated process.

[27].Mitsubishi Heavy Industries of Japan [28] has been using fuzzy logic to control the weld- line tracking by a robot. The sensor signals here are the horizontal and vertical displacements of the welding point.

2.4 Fuzzy Logic Microprocessor related Literature Review

Fuzzy logic has become a universal technology and its application ranges over the entire spectrum of electronic control. The development in software tools enables us to implement fuzzy logic system on almost any target hardware, ranging from low cost 8-bit micro controller that have only few bytes of RAM and few hundred bytes of ROM upto distributed process control system that uses multiple 64 bit RISC processors.

2.4.1 Dedicated Fuzzy Processor

Initial implementation of fuzzy processor came from Japan [29], analogous to discrete logic implementation, the fuzzy processor uses “fuzzy logic gates” for effective implementation. Although they had added advantage of being faster than standard computer processing fuzzy logic but they were not successful, as for every change in rule required change in hardware.

2.4.2 Digital Fuzzy Processor

They were digital processors where conventional instruction set is replaced by a fuzzy logic instruction set. They only supported fuzzy logic computation. The fuzzy processor connects to the host master control unit by either dual ported RAM or a serial communication.

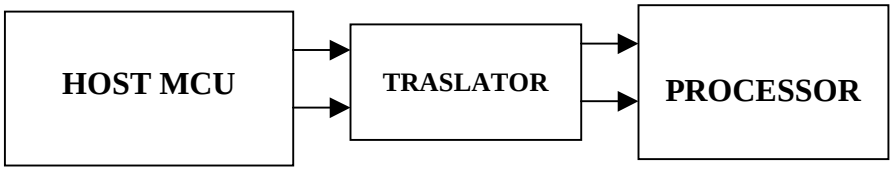

Fig 2.1 Relationship of Processor and MCU

They are not used in industrial application due to the fact that a two-chip solution is both expensive and difficult to design. In addition, programming inter-processor communication between host MCU and fuzzy processor is complicated and requires code and interrupt channels on the host MCU.

2.4.3 Integrated Fuzzy Processor

Integrating the fuzzy processor functionality within standard micro controller, as reported in [30] semiconductor manufactures integrate complete fuzzy logic coprocessor units on the same chip with a micro controller. This has an advantage that fuzzy logic coprocessor and the MCU can work simultaneously. The communication overhead is much smaller compared to the two-chip solution because the fuzzy logic coprocessor and the MCU can work simultaneously and can use the same register for the exchange of the data.

Nowadays more effort is being put for developing dedicated hardware [31]. The current fuzzy logic compilers generate assembly or C code either from fuzzy logic programming or graphical design environment. Fuzzy logic compiler analyses the structure of fuzzy logic system at compile time and pre structures it. From a performance point of view even the fastest fuzzy processor today is not much faster than a micro controller or DSP (direct serial processor) [16]. Even if faster fuzzy processor becomes available, a software solution on a micro controller will be sufficient and most cost effective for the

vast majority of control application. A different solution is to enhance the instruction set of a standard MCU. The easiest way is to just extend the macrocode and use the same ALU architecture. This speeds up fuzzy logic computation up to three times, depending on the MCU architecture. The macrocode extension also compacts the code of a fuzzy logic system about two times, depending on the size of the fuzzy logic system. The integration of a fuzzy logic coprocessor as a macro cell to an existing modular MCU has already been developed by a number of manufacturers. This extension does not require redesign of the MCU. The ALU and the instruction extension have greater potential in the long run, since it delivers a better performance-price ratio. However, only new micro controller design will feature it.

2.4.4 Programmable Logic Controller

Programmable Logic Controllers (PLC) are a common hardware platform in industrial automation and process control. PLC's have standard interfaces to sensors and actors, are rugged for industrial environment, and support standard field buses. Most PLC’s use specialized programming languages such as ladder logic, a semi-graphical way to represent a control algorithm. As stated by T Hogener [32] not all PLC’s support programming in C. Small PLC’s only support ladder diagram, instruction lists, structures text, function block diagram or sequential function charts conforming to the international norm IEC1131.

2.4.5 Fuzzy Logic Implementation on Workstation and PC

There are number of application which are currently using PC or workstation not

only as development platform but are also target platform. As reported by Novak [33] compared to implementations on microcontrollers , PC .

2.4.6 Fuzzy sensor related literature review

A lot of advancement and research has been done in field of development of sensors that can be easily interfaced with fuzzy logic based systems. As reported by Okey [34] Honeywell Corporation employed fuzzy logic controller for their system. The microwave cloth dryer was equipped with an outlet temperature sensor and humidity sensor. The sensors signals and user inputs forms the five inputs variable for the fuzzy logic system.

In a study conducted by German car manufacturer [19] on a model car having a motherboard of a 12 Hz notebook 386 PC control and interface board that drive actors and sensors. The ultrasound distance sensor for track guidance and infrared sensors in every wheel for speed. Stating further the fuzzy logic system interprets the current situation and directs the vehicle after analysis. The low cost ultrasound sensors have been used in this study instead of CCD cameras and picture recognition techniques.

As reported by Davis [35] the Ford Motor Company of U.S used a number of sensors specially designed and developed for heating, ventilating and air-conditioning (HVAC) control in cars. The main goal was to make vehicle comfortable for occupants. Typical sensors of HVAC include a car interior temperature sensor, ambient temperature sensor, and sun heating load sensor, humidity sensors and others.

On the other hand Matsushita Corp. of Japan uses fuzzy logic for a multi sensor array that analyses gas mixes. Each sensor detects a group of gases, hence most gases are detected by more than one sensor. Using different non-linear characteristics of the sensors,

the fuzzy logic data analysis system differentiates and quantifies the ingredients of a gas mix.

2.5 Conclusion

Fuzzy Logic and microprocessor thus represent the integrated technology environment, which can be used for products with multiple functionality and advanced features. The use of fuzzy logic in products can make them user friendly and the vague or imprecise values can be easily interpreted for development of system.

Chapter 3

SYSTEM DEVELOPMENT

For this research the main objective is to develop integrated software with enhanced capabilities of integrating the Handy Board® microprocessor so as to transfer the fuzzy design intent of the user into the system. The system would be based on number of software modules for its implementation. The following pages will discuss in details the various aspects in the software and the hardware design of the integrated system

3.1 Graphical User Interface

The whole scenario and the program development and execution for this research is based on the input from the user which finally gets transformed to the system code. The input from the user would basically be written to a text file from which the main program reads the values. The main features of the software are as follows.

1. Selection of the input and output sensors to be interfaced with the microprocessor.
2. Define the membership function to be adhered to for each sensor.
3. Define the ranges for the membership function
4. Define the limits for each membership functions.
5. State the rules for the interfacing of the input values.
6. Repeat the process for other input and output sensors.

An example text file can be written as:

// Specify all the input and output sensors

Temperature 1 3 0 20 50 50 75 100 100 130 200

Motor 1 3 0 20 50 50 75 100 100 130 200

Beeper 1 7 0 4 10 10 12 30 30 35 40 40 45 70 70 80 100 100 120 150 150 165 200

// Rule format: Sensor Name Order Port State Condition Clause

// Analog: Order Port State Condition Clause

// Beeper: Order State Duration M sleep Repeat

// Motor: Order State Direction // direction: 0 – forward 1 – backward

// Dummy: order

3.2 User Interface

Dynamic Windows environment would be used for implementing the execution of the program developed. Visual Basic 6.0® which is known for its event oriented approach, would be the best platform for development of the front end in which user is required to enter requirements for rules and conditions. Since it also supports objects linking, embedding and dynamic data exchange which further assists the user for real time behavioral aspects. A number of forms are developed for entering the user requirement. A number of control points buttons for switching between various design inlays of the system would be used for successful interpretation of user requirement and available conditions. Pull down menus would also be attached for some blocks giving user the flexibility for choosing between a number of options.

3.3 Development of C++ source code

The C ++ source code developed for fuzzy logic implementation has mainly to take care of the output and finally defuzzification for the system. The code also has to account for the real time transformation of data to the required design intent of the user to

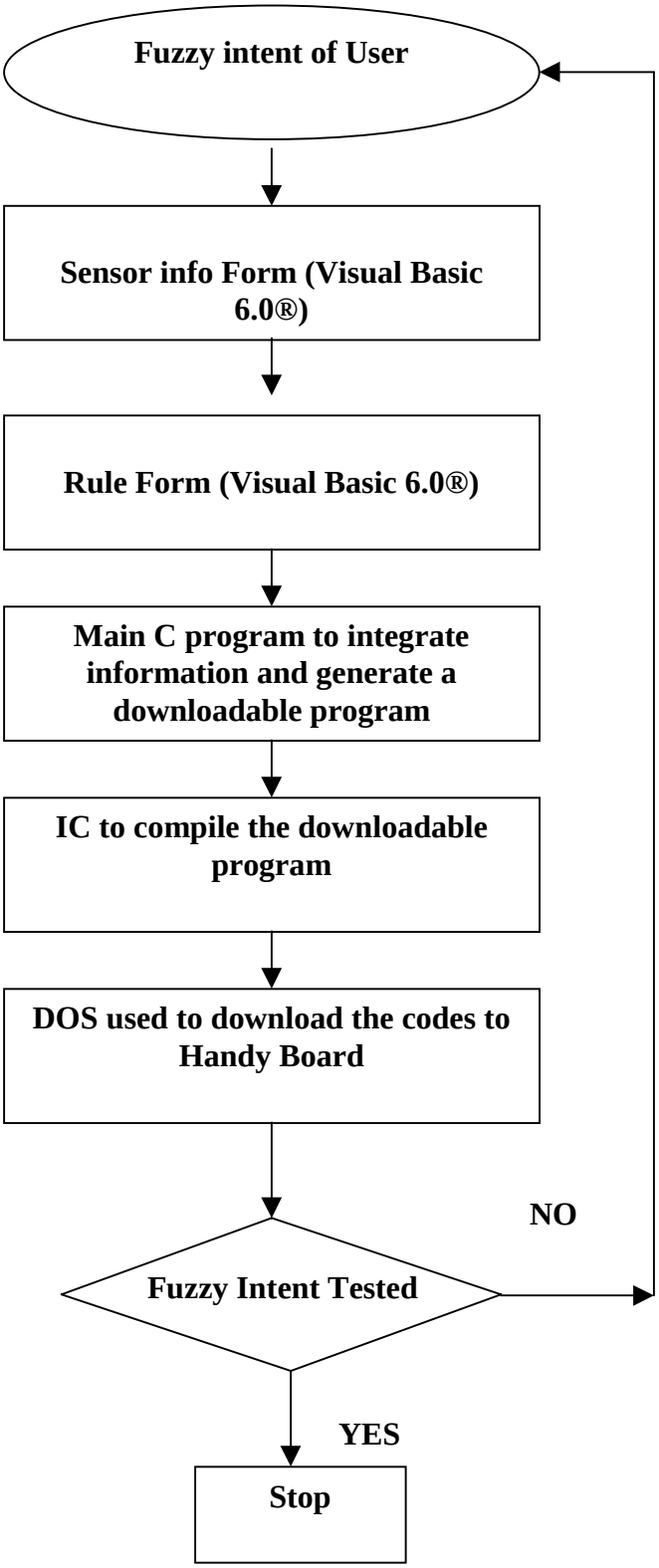

Fig 3.1 Components of the System

be converted into downloadable code for Handy Board® . The code basically fractions down to number of classes for effective implementation.

The program is basically divided into a number of subprograms for effective implementation. The main program takes care of the data to be read from the sensors. The program has been developed for triangular membership function for current use. It can be further extended for other type of trapezoidal, sinusoidal function. The header file was written for the sensor to define characteristics of sensors. The values that are entered by the user from Visual Basic® front end are written to the text file. When the main program is run, the program automatically reads these values and a downloadable program is generated.

The downloadable program basically calculates the fuzzified value after getting the input from the sensor, passing this fuzzified value to a number of rules specified by the user. An output value is generated depending upon which of the rules get fired and these values are then passed to defuzzification section of the program. A crisp value is generated by the defuzzification, which is given for the output of the program.The code of the program is enclosed in appendix for evaluation.

3.4 Data Entry

All the relevant data concerned with the program execution should be entered by the user in forms developed for the same in Visual Basic 6.0®. Incorporating combo boxes provided for different types of sensors, the membership functions and range value options of 3,5,7 get stored in a text file from where the values are then read by C source code for

execution. Some data files have to be generated as the user inputs the values for the membership functions and their functional limits.

3.5 Analog Sensors

The analog sensors can be plugged in the ports labeled as analog input 0 to 7 of the Handy Board®. The values of these analog sensors are in the range from 0 to 255. The input readings that are gathered from the environment are passed to the C++ program to work in real time environment. Basically, all sensors take readings from 0 to 255 for their working. The analog sensors planned to be used for the research include potentiometer and motors.

3.6 Forms

A couple number of forms have been developed in Visual Basic 6.0® to facilitate the development and execution of the software. The forms are designed and developed for easier navigation and control. Several text boxes, option buttons, command buttons, combo boxes are placed on the forms. Codes are written for different buttons to carry out desired functions. The data entered by the user in fuzzy terms in different forms is saved into text file. The forms developed are as follows:

1. Thesis Form
2. Main Form
3. Sensors Form
4. Rule Form

3.6.1 Thesis Form

The first form is just the display of Fuzzy Expert System consisting of two command buttons allowing the user to Proceed or Exit. The part of the code is shown below.

```
Option Explicit
    Private Sub cmdexit_Click()
    End
    End Sub
Private Sub cmdproceed_Click()
    Form2.Show
End Sub
```

3.6.2 Main Form

The main form allows the user to execute a couple number of functions including

Information to be inputted to fuzzy forms (input forms for sensors , rule form)

Run C program for generating downloadable code.

View source code (so as to remove errors in the program).

Compile the code (using Interactive C - IC compiler).

Download to Handy board®.

3.6.3 Sensor Form

The input forms for the sensor allows the user to select the type of sensor, the port in which the sensor is to be plugged as well as giving the option to decide the range in which the sensors are to work. As soon as the range is selected the form has the feature to input the data corresponding to that attribute. It allows the user to input only numeric

values in between 0 to 255 because the analog sensors have a range from 0 to 255. For Example if option 3 is selected by the user, the allowable entries correspond to low, medium and high, for which the values are inputted in the text boxes (0-255). The sample code for the option which does not allow values less than zero and greater than 255 is as follows:

```
Private Sub Text1_Change()
        If Val(Text1.Text) < 0 Then
        MsgBox " Value less than zero can not be entered"
ElseIf Val(Text1.Text) > 255 Then
        MsgBox " Value more than 255 cannot be entered"
Else
        Text1.Text = Val(Text1.Text)
End If
End Sub

Private Sub cmdback_Click()
        Unload Form2
End Sub

Private Sub cmdexit_Click()
        End
End Sub

Private Sub cmdruleform_Click()
        Form3.Show
End Sub
```

3.6.4 Rule form

The rule form allows the user to enter the rules in the fuzzy terms for input and output sensors. It also shows at the bottom number of rules entered. The software does not allow the user to enter more than seven rules and less than one rule. The counter will show the number of rules entered .The sample rule and code for this form is as follows :

Sample Rule:

If potentiometer 1 is low and potentiometer 2 is medium

Then

Motor 1 is medium , motor 2 is low and motor 3 is high .

```
Private Sub cmdexit_Click()
End
End Sub
```

The code for the operation of the counter is below:

```
Private Sub cmdSave_Click()
    Static counter As Integer
    counter = counter + 1
    If counter > 7 Then
    MsgBox " maxm rules entered "
Else
    Print counter
End If
End Sub


Private Sub Combo1_Change(Index As Integer)
Dim i As String
i = ""
MsgBox " u have not entered anything "
```

End Sub

3.7 Viewing of C ++ source code

Any text editor can be used to view C source code finally generated by the program. All the input parameters and the output values on the basis of the crisp values that have been obtained after defuzzification of the rules are stored in this program. The generated source code is stored with extension .C and can be edited in DOS with command “edit filename C”.

3.8 Compiling of C ++ source code

The generated C source code of the program is compiled using an IC compiler for Handy Board® compiler. The compiler resides in the directory “C:\IC\libs. The fuzzy behavior of the associated program that the handy board is capable to execute are copied on to the directory “C\IC\libs”

The compiled program is then downloaded to the Handy-board®.

3.9 Downloading the Program

One end of the modular phone cable is first plugged into the phone jacked socket of the handy board while the other end goes into the printer port of the computer.If the p- code is already loaded in the handy board the procedure outline below can be skipped else

- Press the STOP button and switch the handy board ON. Only the top panel of the LCD will be visible. This is the P- code-downloading mode.
- From C :\ IC directory type “dl pcode_hb.s19”

- Follow the instructions on the computer and reset the board.
- Handy board® is now ready and programs can now be downloaded.

Once p-code has been installed, the following commands will download the programs to the Handy board ®

- From C:\IC type “IC”

  This download all library files and come to C: prompt. To differenciate between this prompt and root directory C: prompt, the former will be mentioned as C: >
- Type the name of the major program file that which has the extension “.lis”. For example to download the Example 1 program file C:> Example 1.lis. Hence, all the associated files will be downloaded along with major program files. To exit from C:> type “Bye”.
- To test the fuzzy intent “switch on” the Handy board®.

3.10 The Design of C++ Program

The C++ program has been developed to read the data stored in a text file. The data saved in text is being entered through the Visual Basic® form. These form are designed and coded to capture the information and save it in required format. The main C++ program has been developed with call commands exclusive to Handy board® by using specific analog and motor function.The entire source code is enclosed in Appendix A.The functions used in the program are briefly below.

Sensor Class

This class is the workhorse for this program. It is responsible for reading the input file with

sensor data, evaluate rule information, generate source code to be compiled and downloaded to the Handy board®, and then write this generated code to an output file.There are a number of functions in this class.

BOOL ReadInput (int argc, char* arg[])

This function checks the validity of command line arguments used to invoke this program. The functions have to be supplied with atleast two arguments. The first argument corresponds to the name of the input file which has sensor and rule information in a specific format. The second argument corresponds to the name of the output file, which will get generated by this program. This output file contains the generated code to be compiled and downloaded to the Handy board®.

Relevant Code:

```
/* Check for validity of command line arguments */
        if (argc < 3)
{
        printf("\tUsage: %s <infile> <outfile>\n\n", argv[0]);
        return FALSE;
}
/* Open the input file and report an error if it does not exist */
m_pInputFile = fopen(argv[1], "r");
if (m_pInputFile == NULL)
{
        printf("Error opening input file %s\n", argv[1]);
        return FALSE;
}
        /* store the output file name to be used later to create the out file*/
m_pOutFileName  = new char[strlen(argv[2]) + 1];
```

strcpy(m_pOutFileName, argv[2]);

The main program then uses some helper functions to parse input, read sensor, read rules, write header information, write body information and some ancillary functions. All of these are described in detail below.

BOOL ParseInput()

This function reads the file, one line at a time till it reaches the end of the file. It parses each line and parses the input. The file has information on all the sensors to be used in the fuzzification process. It then specifies a set of rules. The rule section is bounded by a line with "begin" tag and a line with an "end" tag. All the sensor definitions occur before the"begin" tag.

All sensor definitions start with a sensor name, fuzzy function type and the number of ranges for that sensor. It checks for validity of fuzzy function type and the number of ranges.We currently support Triangular (1) , left tail (2) and right tail (3) fuzzyfunctions and only 3, 5 or 7 as number of ranges. One it determines that it is a valid sensor definition, it parses the entire line to read range information and fill up an internal structure (Sensor m_pSensor) to store sensor information.

Relevant Code:

/ Read properties of an analog sensor */*

retValue = sscanf(cLine,"%s %d %d", cSensorName, &iFuzzyFuncType, &iNumRanges);

/ We currently support only Triangular, left tail and right tail fuzzy functions */*

```
if (iFuzzyFuncType != 1 && iFuzzyFuncType != 2 && iFuzzyFuncType != 3)
{
   return FALSE;
}

/* We currently support only 3/5/7 number of ranges */
if (iNumRanges != 3 && iNumRanges != 5 && iNumRanges != 7)
{
   return FALSE;
}
```

After reading all the sensor information, remaining input file is processed to read rules by ReadRules() function.

BOOL ReadEntireSensor(char* pLine, int iNumRanges)

This function takes a line as an input, parses it to scan information about a sensor and stores it in an internal strcuture (Sensor m_pSensor) to store sensor information. Depending on the number of ranges in the sensor, it reads three, five or seven values along with the sensor name and the fuzzy function. It then checks if the sensor is an input or an output sensor. We currently consider “Motor” as output sensors; all other sensors are considered input sensors. We store this information in a "Boolean" variable in the Sensor structure.

Relevant Code:

```
if (strcmp(strlwr(m_pSensor[m_iNumSensors].m_pSensorName), "beeper") == 0)
{
      m_pSensor[m_iNumSensors].m_bIsBeeper = TRUE;
}
else if (strcmp(strlwr(m_pSensor[m_iNumSensors].m_pSensorName),
"motor") == 0)
```

```
{
    m_pSensor[m_iNumSensors].m_bIsMotor = TRUE;
}
else
{
    m_pSensor[m_iNumSensors].m_bInputSensor = TRUE;
}
```

BOOL ReadRules(void)

This function reads rule information from the file. End of rule section is marked by sensor by checking it against the array of sensors read earlier. Once the sensor name is verified, the line is passed to ReadEntireRule () to parse it for the rule information.

Relevant Code:

```
/* get the sensor name */
sscanf(cLine, "%s", cSensorName);

/* are we done reading rules? -- check for "end" tag */
if (strcmp(strlwr(cSensorName), "end") == 0)
{
   bIsDone = TRUE;
   break;
}

/* find if this is a valid sensor name */
iCurrentSensor = FindSensor(cSensorName);
if (iCurrentSensor == -1)
{
   return FALSE;
}
```

/ read entire rule */*

ReadEntireRule (cLine, cSensorName, iCurrentSensor);

BOOL ReadEntireRule (char* pLine, char* pSensorName, int iSensorIndex)

This function takes a line as input, parses it to scan information about a rule and stores it in an internal structure the rule information (Rule m_pRule) .The line is scanned for different information based on whether the sensor in the rule is an input sensor, a motor or a beeper. All the rules start with an Ordinal number. The ordinal number is used to specify scope of a given rule. An input sensor has Port, State, Condition and Clause as arguments. A Motor is described by specifying State and Direction. All this information is stored in the Rule data structure. It also checks for the validity of the rule.

BOOL WriteHeader ()

This function opens the output file for writing. It writes the necessary header information to this output file. It writes some "#define"s and some "#include"s necessary for the output code. It also writes some helper functions to be used later in the "body" part of the code. These could be fuzzymotor(), CalculateMembership(), Calculate_righttail_Membership(), Calculate_lefttail_Membership(), DeFuzzify() functions.

Relevant Code:

/ print the include info */*

fprintf(m_pOutputFile, "# <mboard.h>\n\n");

/ define WANT_BEEPER constant if a beeper is one of the output sensors*/*

if (m_bWantBeeper)

{

```
        fprintf(m_pOutputFile, "#define WANT_BEEPER 1\n");
    }
        /* define WANT_MOTORS constant if a motor is one of the output sensors*/
    if (m_bIsForwardMotor || m_bIsBackwardMotor || m_bWantMotor)
    {
        fprintf(m_pOutputFile, "#define WANT_MOTORS 1\n");
    }
```

BOOL WriteBody ()

This function is where the bulk of work is done for writing out the generated code. It writes the "main" function. It starts off by writing out some temporary variables that will be used in the main() routine.It maintains the tab count to properly format the output. It runs through each rule and generated required code for the given rule. For an input sensor, it prints a line to read the input from the sensor. It then calculates the appropriate membership function depending on the fuzzy function type denoted by m_iFuzzyFuncType. For an output sensor, it makes a call to Defuzzify () and then passes the defuzzied value to the output sensor by making a call to fuzzymotor () function. It also does a bookkeeping of the number of tabs to be inserted in the beginning of each line, of the closing braces for the "if" statement etc.

The program is coded to write the main section of the program, which performs the calculation for fuzzification, and defuzzification of the values, which the program reads while in operation. The functions in the body of the program calculate the membership values depending whether it is right tailed or left tailed triangular function. A number of calculations are performed taking into account the range in which the value read from sensor is falling as per the rule specified by the user.

3.11 Conclusion

The chapter highlights the various outlines of software's required for development of fuzzy logic based rapid prototype of the system and its related features. The user-friendly front end in this system will facilitate the use of system even by the person not used to fuzzy logic basics.

Chapter 4

SYSTEM IMPLEMENTATION

4.1 Installation of Software

The software for the thesis is in form of an executable format To install the software the user has to enter a " A:\ Fuzzy Thesis" in DOS environment, or open the file in windows explorer and click on file name Fuzzy Thesis.exe in addition to this software for the purpose of compiling and downloading of generated C++ source code. The IC compiler of interactive C has to be installed. This compiler is installed in C:\IC\libs .Microsoft Visual Basic 6.0® and Microsoft Visual C ++ 6.0® should be the integral part of the system on which this software should be run .

4.2 Executing the software

The windows Visual Basic 6.0® environment is necessary for running this program. The forms are all developed in Visual Basic 6.0® and Visual C++® is used for compiling and generating the code for a downloadable C program for the handy board .The software can be run by typing "C:\IC\libs". In the Run section of the start menu .the main screen of the software is shown in figure 4.1 .

Fig 4.1 Opening Screen For the software

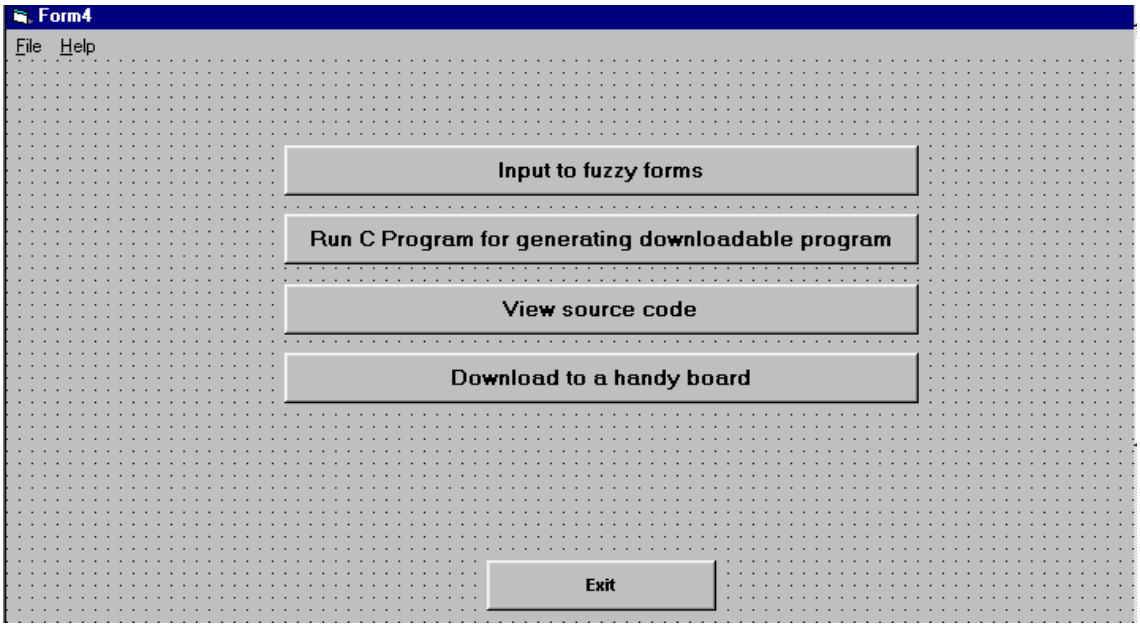

Fig 4.2 Main Screen of the software

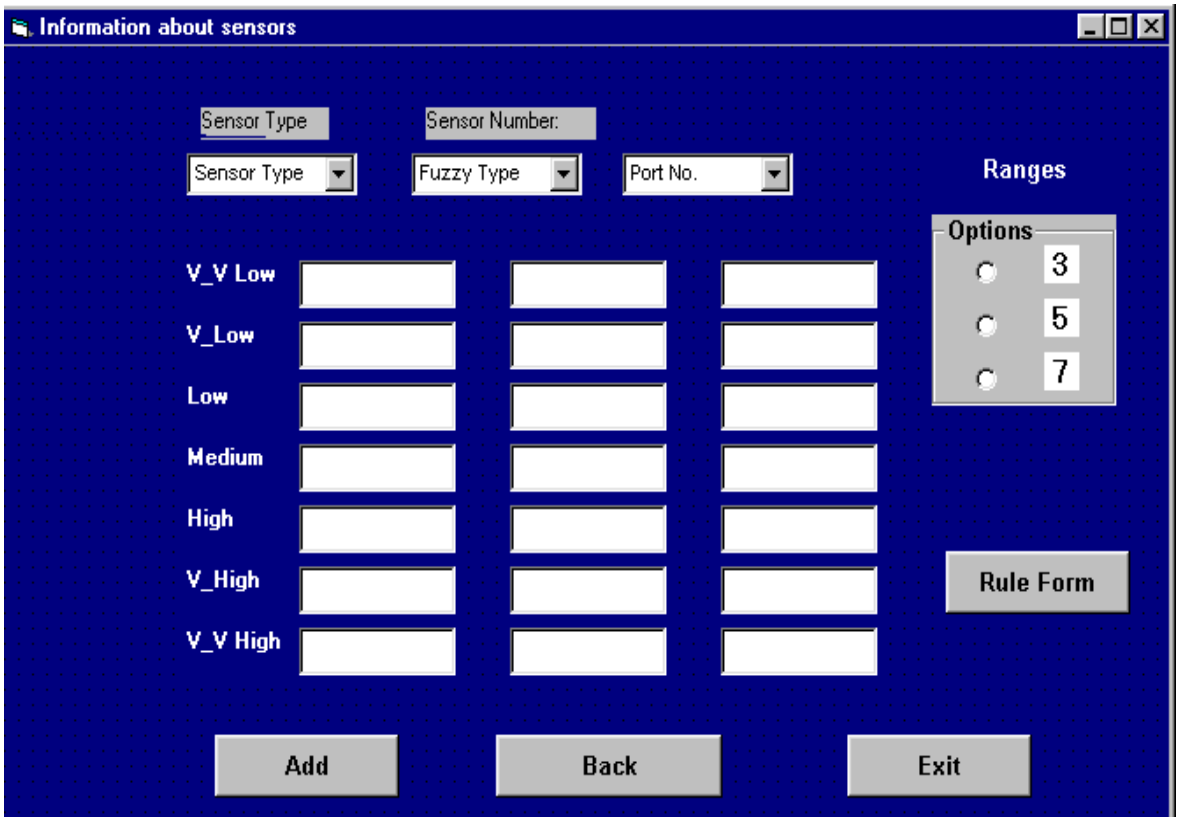

Fig 4.3 Sensor Info Form

Main form (Fig 4.1) has two buttons for " Proceed" and "Exit". The Proceed enables the user to move to the next form while Exit button takes the user to the windows screen.

The second form (Fig 4.2) allows the user to perform various functions with the software. The command button titled " Input to Fuzzy Forms" takes the user to the respective forms.

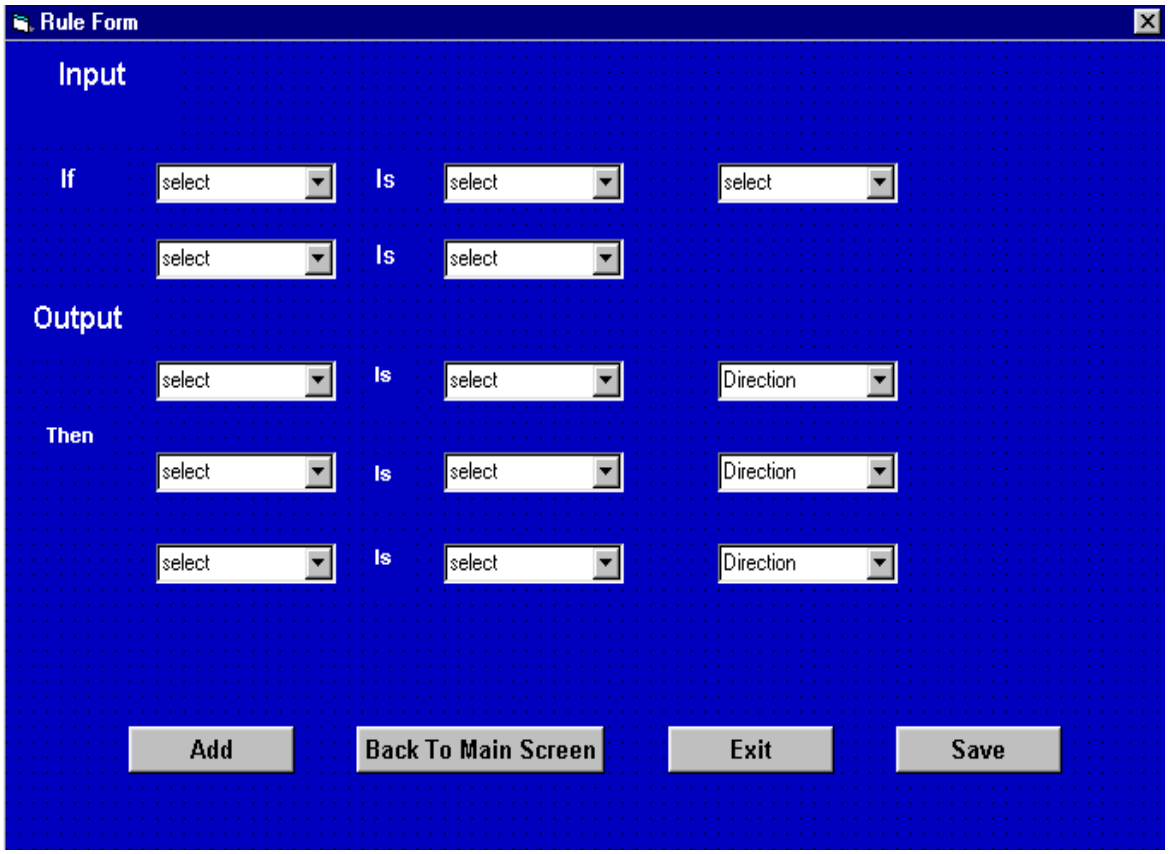

Fig 4.4 Fuzzy Rule Form

The Sensor Info Form (Fig 4.3) and Fuzzy Rule form (Fig 4.4) allows the user to enter various requirements regarding sensors and Fuzzy rules .The other options come into picture after the user has entered all the requirements into a form . After the data has been saved in a text file the user can run the C program by pressing the command button "Run C Program for generating downloadable program". The command button for "View source code" on the main form gives the user an option to view source code so that the intents can be cross checked with the results . The next command button "Compile the

code" facilitates compiling of C++ code before it is downloaded to the Handy board® . The last command button "Download to Handy Board®" allows the user to download C program on to the Handy board® for execution of the fuzzy intent .

4.3 Data Input for Fuzzy Intent

Two examples are explained below to cover the fuzzy behavioral analysis for the user . The examples are shown with calculations required to execute the fuzziness with real time control. Examples are as follows :

The fuzzy intent of the user is

Example 1

If Potentiometer 1 is low

And

Potentiometer 2 is Medium

Then

Motor1 is Low

The following requirements are entered in Sensor Info Form and Rule Form. As seen from the rules the user is only dealing with low, medium and high for potentiometer consequently he chooses option 3 which allows him to enter his values for low, medium high (each of these takes three values since triangular function is being used). In addition he also enters the information regarding the port number where he plugs his input sensor (which is port number 1 in this case) and the sensor name (which is potentiometer 1 in this case). The second form is the Rule Form, which allows the user to enter the rules in the form in the required format. For input sensor the rule 1 says potentiometer 1 is low and

potentiometer 2 is medium and for output sensor it says Motor 1 is low which gets entered into the rule block. After the user presses the Save command button the entire rule gets saved in text file in a required format for the C++ program to read it. Similarly, rule 2 and rule 3 gets entered into the Rule Form. The example text file is shown below.

***Example 1 text file*:**

```
// Specify all the input / output sensors here
Potentiometer 1        1 3 0 20 50 50 75 100 100 130 255
Potentiomater 2        1 3 0 20 50 50 75 100 100 130 255
Motor 1                1 3 0 30 50 50 80 100 100 180 255
// done with sensor specification
// begin specifying rule in the following format
Rule Format : Sensor Name Order Port  State Condition
Begin
// Rule 1
Potentiometer 1        1       1       0       3       1       0
Potentiometer 2        1       1
Motor 1                0       1       1
Dummy                  1
```

The information entered in the text file is read by C++ program for generating a downloadable code. All the information for the input sensor or output device is explained as below.

Potentiometer1 1 3 0 20 50 50 75 100 100 130 255

The digit 1 in the start informs the program that sensor is analog. The next digit 3 communicates to the program the number of ranges the sensor is going to operate, in this case it happens to be Low, Medium and High for Potentiometer 1. The other digits are in

pairs of three. Consecutively, the first three (0 20 50) correspond to Low range. The next three digits correspond for Medium range and last three for High range.

The rules specified by the user are also saved in similar fashion. The line starts with sensor name. The second digit is for opening up of loop for this rule condition. The third digit gives the Port number to which either the input sensor or output device is plugged. The fourth digit gives the state of the sensor whether it is V_V_Low, V_Low, Low, Medium etc. Since the program is generic it can even work for digital sensors as well, and the fifth digit gives the information regarding the condition for two sensors joined either by 'AND' or 'OR'. The default is 0 with AND taking digit 1 and OR depicted by digit 2.

As we see from the fuzzy intent of the user which dictates that Motor1 to be low when Potentiometer1 is low and Potentiometer2 is medium. Thus, with different settings of Potentiometer the corresponding effect can be observed in the output of the Motor speed. The fuzzification and defuzzification of the values are being performed by the C++ program for generating membership values and final crisp values for output to the Motor speed. The following graph shows the fuzzification and defuzzification process for example one and how the crisp value gets transferred to the output device. As we can see from the graph the Potentiometer1 corresponds to low range and Potentiometer2 corresponds to Medium range. The fuzzified values from both the graphs are then compared in the program and the minimum of both is passed for the defuzzification. Thus, the second graph for the motor1 then takes this fuzzified value and calculates a crisp value in the range where motor1 needs to be operated. In our example the motor1 is operating in the low range.

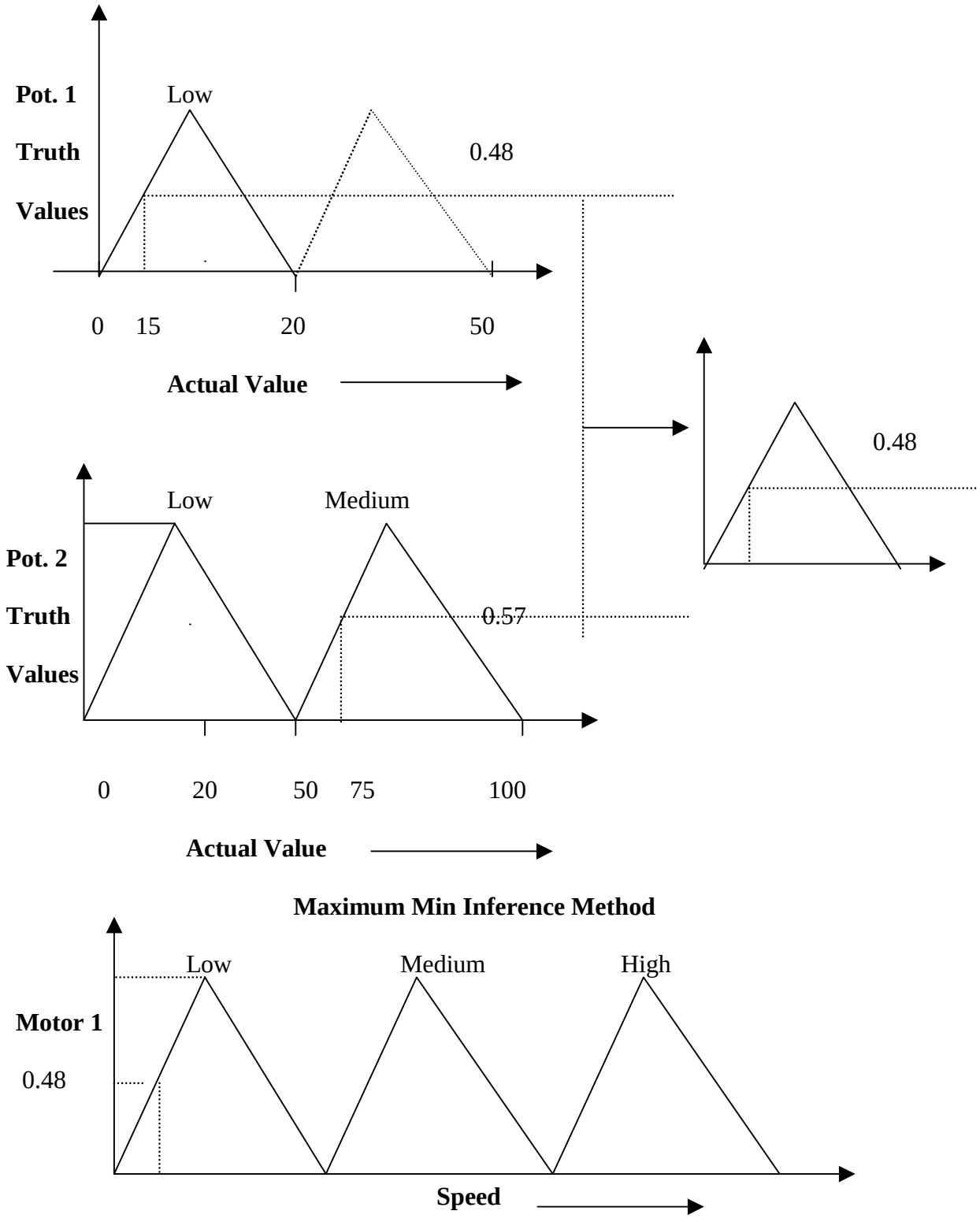

Fig 4.5 Fuzzification and Defuzzification Example 1

Example 2

If Potentiometer 1 is V_Low

And

Potentiometer 2 is Medium *Then*

Motor 1 is V_High ,Motor 2 is Medium and Motor 3 is V_Low

The above-mentioned requirements are for two sensors (potentiometer 1, potentiometer 2) are entered consequently. The information about the first sensor is for Range (option 5) which takes all the values except V_V High and V_V Low (three in each case as it is a triangular function) . The information about the second sensor is for Range (option 3)since the user selects it in the Medium range. The user then enters the information about the output device (Motor 1, Motor 2, Motor 3) as per the ranges specified in the rules which are 5,3,5 respectively . The user than moves to the Rule Form where he enters the exact rules as mentioned above. The maximum numbers of rules that the user can enter are 5. After all the rules have been entered the user presses the Save command button the entire rule is saved in a text file for a C++ program to read it and run the main program to generate a downloadable C++ program for a handy board. the example text file is shown below .

Example2 Text file:

Potentiometer 1	1 5 0 10 20 20 30 50 50 100 150 150 175 200 200 225 255
Potentiometer 2	1 3 0 10 20 20 30 50 50 100 150
Motor 1	1 5 0 20 50 50 75 100 100 125 150 150 175 200 200 225 255
Motor 2	1 3 0 20 50 50 75 100 100 130 200
Motor 3	1 5 0 15 30 30 45 50 50 100 150 150 175 200 200 225 255

// done with sensor specification

// begin specifying rules in the following format

Rule Format Sensor Name Order Port Start Condition

// Rule 1

Potentiometer 1	1	2	1	1
Potentiometer 2	1	3	3	0
Motor 1	2	1	5	1
Motor 2	3	2	3	1
Motor 3	3	3	1	0
Dummy	1			

The sensor ranges are as follows for which data values are entered.

1 V_V Low

2 Low

3 Medium

4 High

5 V_High

After all this information is entered the C++ program reads the text file and a downloadable code is generated. The fuzzy intent of the user in this example is as long as both the conditions for Potentiometer 1 is V_Low and Potentiometer2 is medium are satisfied then Motor1 would run at very high speed. Motor2 should run at medium speed and Motor 3 would run at V_Low speed. We can see in the example text file that the information for different sensors entered corresponds to the required output by the user. The information for the Potentiometer 1 has five range values, similarly Potentiometer 2,Motor1, Motor2, Motor3 have also the information in required format. If now the rule base by the user gets changed to one below then the program would respond if either of the conditions are satisfied.

If Potentiometer 1 is V_Low

OR

Potentiometer 2 is Medium *Then*

Motor 1 is V_High ,Motor 2 is Medium and Motor 3 is V_Low

Thus, the user intent can be repeatedly tested and retested for changing fuzzy rule base and program responds to the same by changing the output as desired.

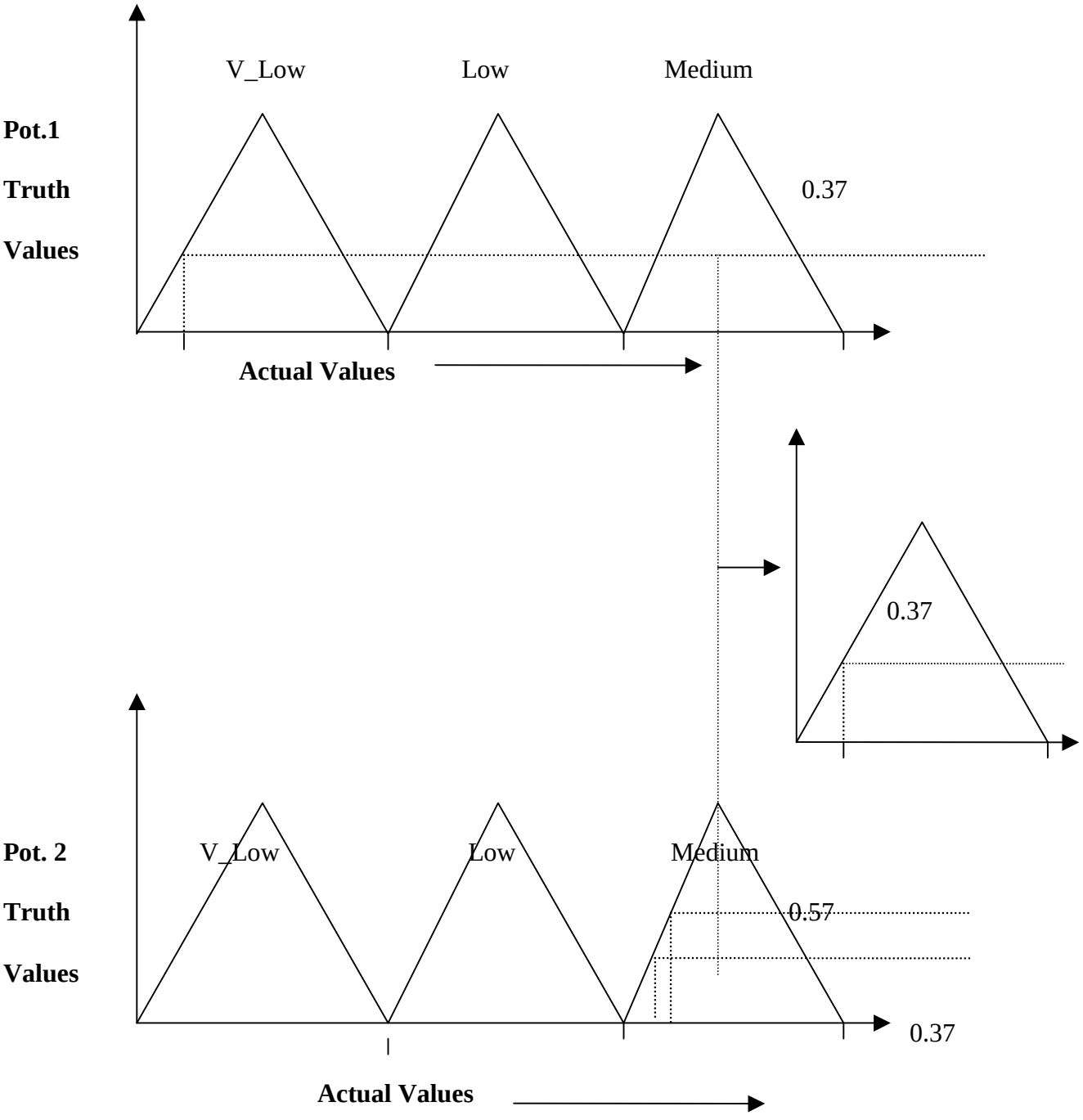

Fig 4.6 Fuzzification of Example 2

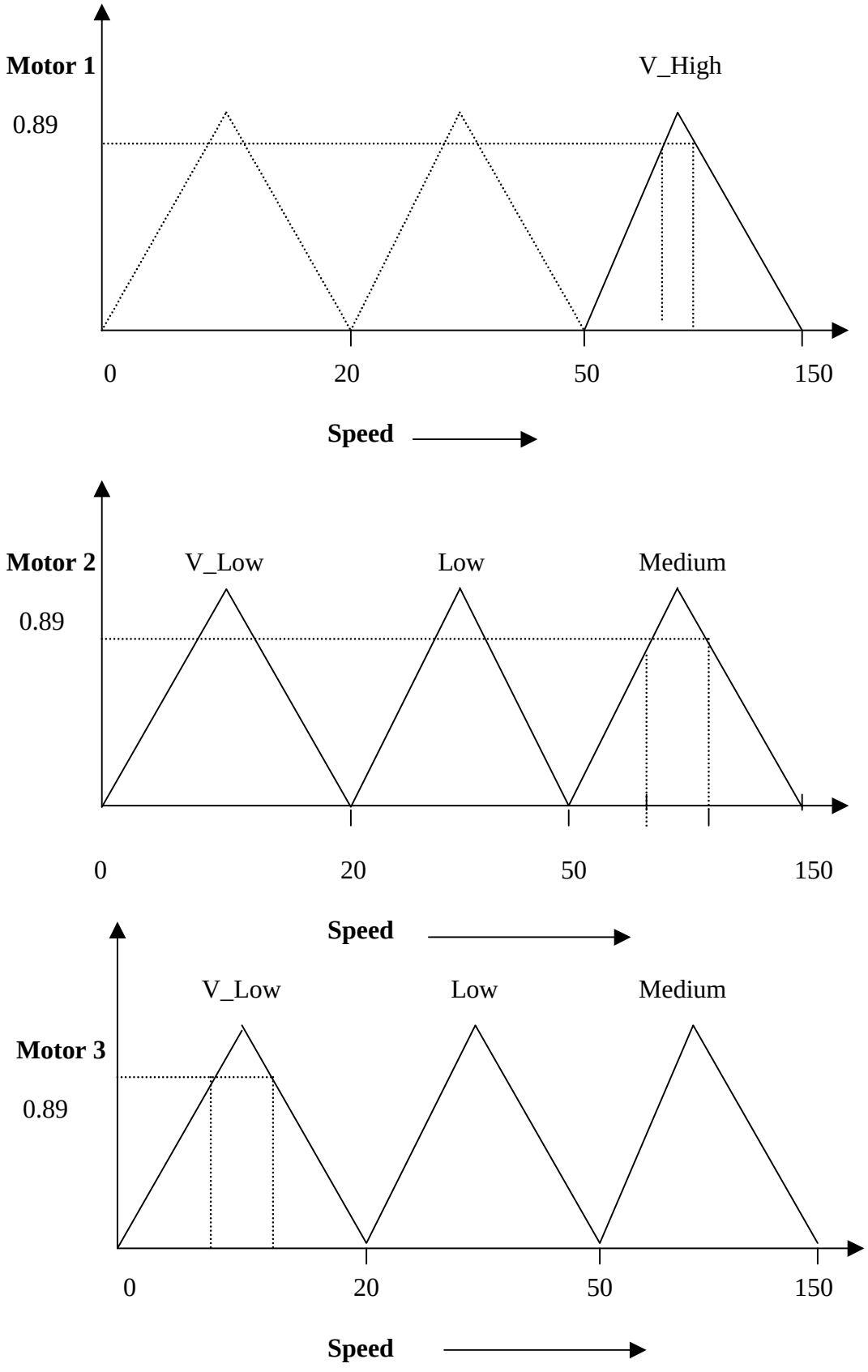

Fig 4.7 Defuzzification of Example 2

4.4 Executing the C++ Program

The C++ program is started by clicking the button " Run C program for generating the downloadable code". The output of the program gets saved into a temporary folder as full path "C:\Temp\output.c".

4.5 Generated C++ source code

The generated C++ source code can be viewed by clicking on " View source code" The source codes for two examples are shown below.

4.5.1 Source code for Example 1

```
void fuzzymotor (int speed, int direction)
{
	int var1[4], var2[4];
	int i;
	if (direction == 0) {
		var1[0] = -speed; var1[1] = 0; var1[2] = speed; var1[3] = 0;
		var2[0] = 0; var2[1] = -speed; var2[2] = 0; var2[3] = speed;
	}
	else {
		var1[0] = speed; var1[1] = 0; var1[2] = -speed; var1[3] = 0;
		var2[0] = 0; var2[1] = -speed; var2[2] = 0; var2[3] = speed;
	}
	for (i=0; i<4; i++)
	{
		motor(1,var1[i]);
		motor(1,var2[i]);
```

```
        }
}
int Calculate Membership(int x, int low, int apex, int high, float* membership)
{
        if (x >= low && x <= high)
        {
                if (x <= apex)
                        *membership = ((x - low) * 1.)/(apex - low);
                else
                        *membership = ((high - x) * 1.)/(high - apex);
                return 1;
        }
        else
        {
                return 0;
        }
}
void DeFuzzify(float fuzzyValue, int left, int apex, int right, int* meanValue)
{
        int x1, x2;
        x1 = left + fuzzyValue*(apex-left);
        x2 = right + fuzzyValue*(apex-right);
        *meanValue = (x1 + x2) / 2;
}
void main(void)
{
        int a[10];
        float fMemberValue[10];
        int iSensorValue = 0;
        int iNumActiveRules = 0;
        float fMembership = (float) 0.;
```

```
	int iValue = 0;
	while(1)
	{
		fMembership = (float) 0.;
		iNumActiveRules = 0;
		iSensorValue = analog(10);
		if (CalculateMembership(iSensorValue, 0, 20, 50, &fMembership))
		{
			fMemberValue[iNumActiveRules] = fMembership;
			iNumActiveRules++;
			DeFuzzify(fMembership, 70, 80, 100, &iValue);
			tone(iValue, 25);
			msleep(10);
		}
		iSensorValue = analog(10);
		if (CalculateMembership(iSensorValue, 50, 75, 100, &fMembership))
		{
			fMemberValue[iNumActiveRules] = fMembership;
			iNumActiveRules++;
			DeFuzzify(fMembership, 50, 75, 100, &iValue);
			fuzzymotor(iValue, 0);
			DeFuzzify(fMembership, 10, 12, 30, &iValue);
			for (int b=0; b < 5; b++)
			}
	}
}
```

4.5.2 Source Code for Example 2

```
void fuzzymotor (int speed, int direction)
{
	int var1[4], var2[4];
```

```
	int i;
	if (direction == 0) {
		var1[0] = -speed; var1[1] = 0; var1[2] = speed; var1[3] = 0;
		var2[0] = 0; var2[1] = -speed; var2[2] = 0; var2[3] = speed;
	}
	else {
		var1[0] = speed; var1[1] = 0; var1[2] = -speed; var1[3] = 0;
		var2[0] = 0; var2[1] = -speed; var2[2] = 0; var2[3] = speed;
	}
	for (i=0; i<4; i++)
	{
		motor(1,var1[i]);
		motor(2,var2[i]);
		motor(3, var[I];
	}
}
int CalculateMembership(int x, int low, int apex, int high, float* membership)
{
	if (x >= low && x <= high)
	{
		if (x <= apex)
			*membership = ((x - low) * 1.)/(apex - low);
		else
			*membership = ((high - x) * 1.)/(high - apex);
		return 1;
	}
	else
	{
		return 0;
	}
}
```

```
void DeFuzzify(float fuzzyValue, int left, int apex, int right, int* meanValue)
{
	int x1, x2;
	x1 = left + fuzzyValue*(apex-left);
	x2 = right + fuzzyValue*(apex-right);
	*meanValue = (x1 + x2) / 2;
}
void main(void)
{
	int a[10];
	float fMemberValue[10];
	int iSensorValue = 0;
	int iNumActiveRules = 0;
	float fMembership = (float) 0.;
	int iValue = 0;
	while(1)
	{
		fMembership = (float) 0.;
		iNumActiveRules = 0;
		iSensorValue = analog(10);
		if (CalculateMembership(iSensorValue, 0, 20, 50, &fMembership))
		{
			fMemberValue[iNumActiveRules] = fMembership;
			iNumActiveRules++;
			DeFuzzify(fMembership, 70, 80, 100, &iValue);

		}
		iSensorValue = analog(10);
		if (CalculateMembership(iSensorValue, 50, 75, 100, &fMembership))
		{
			fMemberValue[iNumActiveRules] = fMembership;
```

```
                    iNumActiveRules++;
                    DeFuzzify(fMembership, 50, 75, 100, &iValue);
                    fuzzymotor(iValue, 0);
                    DeFuzzify(fMembership, 10, 12, 30, &iValue);
                    for (int b=0; b < 5; b++)

                }
        }
}
```

4. 6 Conclusion

This chapter gives the detailed description of how to use the developed software. The various features have been explained so as to help the user to execute in most friendly fashion without even knowing Visual C++®, Visual Basic 6.0®.

Chapter 5

CONCLUSIONS AND FUTURE WORK

The field of fuzzy logic is emerging fast as the technological solutions to future product and system development. As world moves to develop system with more decision making power, synergistically integrating different technologies from all disciplines, the fuzzy logic turns out to be smarter discipline to design and develop advanced products.

The major constraint with present version of Handy Board® [V1.2] is with the number of sensor ports and motor driven. Well infact for fuzzy logic implementation the microprocessor speed turns out to be a major bottleneck. As number of rules increase the microprocessor takes longer time to go through each and every rule and generate an output with increased (microprocessor) speed this could be drastically reduced. The interactive C version that is presently being used does not support multidimensional array sand structures. This adds restrictions in programming, as complicated program may require the use of advanced features of arrays and structures. In addition to that pre – processor statements like DEFINE are not possible with present version. However Newtonlabs have been working on the latest version of Interactive C 3.1 and are trying to incorporate the same.

With neural networks, the contribution of various behaviors can be estimated by giving weightage. Instead of one behavior dominating the final outcome these advanced programming techniques are to be computed in real time with minimal error and require advanced programming features which are not available with the present version of Interactive C.

5.1 Conclusions

A Fuzzy Expert System was developed using Visual Basic 6.0® as user interface and utilizing Handy Board® to control the behavioral aspect of output devices (motors). The behavior algorithm was coded using C++ programming language.The system facilitates generation and testing of Fuzzy intent of user by downloading the program into the handy board. Then the fuzzy logic could immediately be verified and changed if required by the as per user's specification.

REFERENCES

1. Zadeh.L.A., “Fuzzy Sets, Information and Control Vol. 8”, 1965, Pg.338–353.

2. Gupta.M.M., Kiszka. J.B. and Trojan.G.M., “Multivariable structure of Fuzzy Control Systems”. IEEE Transactions on Systems, Man and Cybernetics, 1986, Pg.94-102.

3. Zadeh.L.A., “ Fuzzy Algorithms, Information and Control, Vol.12”, 1968, Pg.94–102.

4. Zadeh.L.A., “ A Rationale for Fuzzy Control, J Dynamic Systems, Measurement and Control, Vol.94, series G”, 1972, Pg.3-4.

5. Zadeh.L.A., “ Outline of a New Approach to the Analysis of Complex Systems and Decision Processes”, IEEE Transactions Systems, Man, and Cybernetics, Vol. SMC-3, 1973, Pg.28–44.

6. Madami.E.H. and Assilian.S., “ A Study on the Application of Fuzzy Set Theory to Automatic Control,” Proc, IFAC Stochastic Control Symp. Budapest, 1974.

7. George.J., Klir / Yuan. Bo, “ Fuzzy sets and Fuzzy Logic”, 1995 Edition.

8. Preprints of the Second Congress of International Fuzzy System Association, Tokyo, Japan 1987.

9. Yasunobu.P. and Miamoto.P., “ Automatic Train Operation by Predictive Fuzzy Control ” in Sugeno (Ed.) Industrial Applications of Fuzzy Control, Amsterdam, New York, 1985, Pg.1-18,

10. Martin.F., “A toolkit for learning: Technology of MIT LEGO Robot Design competition”, Proceeding of the workshop on mechatronics education, July ‘1994 , Pg.57–67.

11. Jones.J. and Flynn.A., “Mobile Robots – Inspiration to implement “, published by A K Peters, Wellesley, Massachusetts, 1993.

12. Mamdami.E.H. and Assilian, S., “An experiment in linguistics synthesis with a Fuzzy Logic controller”, International Journal of Man Machine Studies, 1975, Pg.1-13.

13. Gupta.M.M. and Qi.J., “Design of fuzzy logic controller based on generalized T-operators”, Fuzzy Sets and Systems, 1991b,Pg.42-44.

14. Becker.K., Kasmacher.H., Rau.G., Kalff.G., and Zimmerman.H.J., “A Fuzzy Logic Approach to Intelligent Alarms in Cardiaoanesthesia”, Third IEEE Conference on Fuzzy Logic, 1994, Pg.2072- 2076.

15. Kawai.H. et al., “Engine Control System, ” Proc. Of the Int’l Conf. On Fuzzy Logic & Neural Network, IIZUKA, Japan, 1990, Pg.929-937.

16. Ikeda.H. et al., “An Intelligent Automatic Transmission Control Using a One Chip Fuzzy Inference Engine,” Proceedings of the International Fuzzy Systems and Intelligent Control Conference in Louisville, 1992, Pg.44–50.

17. Takahashi.H., Ikeura.K., and Yamamori.T., “5- Speed Automatic Transmission Installed Fuzzy Reasoning”, IFES ’ – Fuzzy Engineering toward Human Friendly Systems,1992, Pg.1136-1137.

18. Sakaguchi.P. et al., “Application of Fuzzy Logic to Shift Schedulign Method for Automatic Transmission”, Second IEEE International Conference on Fuzzy Systems, ISBN 0-7803-0615-5,1993, Pg.52-58.

19. Von.Altrock C., Krause.B., and Zimmermann.H.J. “Advanced Fuzzy Logic Control of a Model Car in Extreme Situations,” Fuzzy Sets and Systems, Vol.48, No.1, 1992, Pg.41-52.

20. Takayama.A., " Automatic Cruising system Using Fuzzy Logic Control, "Journal of Japan Society for Fuzzy Theory and Systems, Vol.3, No.2, 1991, Pg.168–169.

21. Colin.Johnson R., "Electronic Engg Times Issue 1064", 1999, Pg. 58.

22. Boone.Mary E., Sales and Marketing management,Vol.151, Issue 2, 1999, Pg77.

23. Bob.Pease., " Electronic Design", Vol.46, Issue 15, June'1998, Pg.137/ IP / IC.

24. Boning.D., "Methodical Tool Bases Design of a Fuzzy Controller for a Granulated Medium Refining Machine, " EUFIT' 93 – First European Congress on Fuzzy and Intelligent Technologies in Aachn, 1993, Pg.1016–1022.

25. Ramaswamy.P., Edwards.R.M., and Lee.K.Y., "An Automatic Tuning Method of a Fuzzy Logic Controller for Nuclear reactor," IEEE Transatons on Nuclear Sciences, 1993, 40 (4), Pg.1253-1262.

26. Chen.H., Mitzumoto.M., and Ling.Y.F., " Automatic Control Of Sewerage Pumpstation by Using Fuzzy Controls and Neural Networks, " 2nd International Conference on Fuzzy Logic and Neural Networks Proceedings, IIZUKA, Japan, ISBN 4-938717-01-8, 1992, Pg.91-94.

27. N.N., " Fuzzy Theory Control of Cold Strip Mill " Tehno Japan, Vol.23, No.3, 1990, Pg.33-38.

28. Shimojima.K. et al., "Sensors integration Utilizing Fuzzy Inference with LED Displacement Sensor and Vision System," Second IEEE International Conference on Fuzzy System, ISBN 0-7803-0615-5, Pg.59-64.

29. Yamakawa.T., Shirai.Y. and Ueno.F., " Implementation of Fuzzy Logic Hardware systems," Transactions of the IEEE, Vol.c-63, 1980, Pg.720-725 and Pg.861-862.

30. Corbin.J., "Fuzzy Logic-Based Financial Transaction Card Security System," Embedded Systems Conference in Santa Clara, 1994.

31. N.N., "Fuzzy Tech 4.0 MCU Edition Manual", INFORM GmbH Aachen/Inform Software Corp., Chicago, 1995.

32. Hogener.J., " Fuzzy – PLC: A Connection with a Future, " EUFIT' 93 – first European Congress on Fuzzy and Intelligent Technologiesin Aachen, 1993, Pg.688-691.

33. Novak.V., " Fuzzy Sets and Their Applications, " Adam Hilger,San Francisco, 1989.

34. Okey.Ch., Foslien.W., and Kesselring.J., " Fuzzy Logic Controls for the EPRI Microwave Clothes Dryer, " Third IEEE International Conference on Fuzzy Systems ISBN 0-7803-11896-X, 1994, Pg.1348-1353.

35. Davis.L.I. et al., "Fuzzy Logic for Vehicle Climate Control", Third IEEE International Conference on Fuzzy Systems, ISBN 0-7803-1896-X, 1994, Pg.530-534.

36. Hewit.J. and King, T., "Mechatronic Design for Product Enhancement", Robotics and Autonomous Systems, 1996, Pg.135-142.

37. Ashley.S., "Getting a hold on Mechatronics", Mechanical Engineering, May'1997, Pg.60-63.

38.Rodrigo. Martínez-Béjar, Hossein. Shiraz and Paul Compton "Spanish Scientific Research Council, Avda. La Fama, 1, C.P. 30080-Murcia, Spain. E-mail: rodrigo@pandora.dif.um.es, 1997a.

Appendix A

Source code for the C Program

```
/* This is main.cpp that is called in the main program
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

#include "sensor.h"

main(int argc, char* argv[])
{
   Sensors  sensor;
   int       retVal  = 0;

   retVal = sensor.ReadInput(argc, argv);

   getchar();

   return retVal;
}

/* This is sensor.h which is called  the main program

                #ifndef _SENSORS_H
                #define _SENSORS_H

                typedef int BOOL;
                #define MAX_LENGTH       1000
                #define MAX_SENSOR_NAME      100

                #define MAX_NUM_RANGES       7
                #define MAX_NUM_SENSORS      8
                #define MAX_NUM_RULES        20


                #define TRUE 1
                #defineFALSE          0

                typedef enum
                {
                   AND = 1,
                   OR = 2
                } CONDITION;
```

```
/*
typedef enum
{
   VVLOW = 0,
   VLOW,
   LOW,
   MEDIUM,
   HIGH,
   VHIGH,
   VVHIGH
} STATE;
*/

struct Sensor
{
   Sensor()
   {
         m_iNumRanges            = 0;
         m_iFuzzyFuncType    = 0;
         m_pSensorName[0]    = '\0';

         m_bInputSensor  = FALSE;
         m_bIsDummy     = FALSE;
         m_bIsBeeper      = FALSE;
         m_bIsMotor       = FALSE;
   }

   int       m_iNumRanges;
   int       m_pLowRange[MAX_NUM_RANGES];
   int       m_pHighRange[MAX_NUM_RANGES];
   int       m_pApex[MAX_NUM_RANGES];
   char    m_pSensorName[MAX_LENGTH];
   int       m_iFuzzyFuncType;

   BOOL    m_bInputSensor;
   BOOL    m_bIsDummy;
   BOOL    m_bIsBeeper;
   BOOL    m_bIsMotor;
};

struct Rule
{
   Rule()
   {
         m_iSensorIndex         = -1;
         m_iOrder       = -1;
```

```
        m_iPort                 = -1;
        m_iState        = -1;
        m_iCondition    = -1;
        m_iClause       = -1;

        m_iDuration     = -1;
        m_iNumRepeats       = -1;
        m_iSleepTime    = -1;
        m_iDirection    = -1;
    }

    int     m_iSensorIndex;
    int     m_iOrder;
    int     m_iPort;
    int     m_iState;
    int     m_iCondition;
    int     m_iClause;

    /* For Beeper */
    int     m_iDuration;
    int     m_iNumRepeats;
    int     m_iSleepTime;

    /* For Motor */
    int     m_iDirection;
};

class Sensors
{
public:
        Sensors();
        ~Sensors();

    BOOL    ReadInput(int argc, char* argv[]);

protected:
    BOOL    ParseInput();
    BOOL    WriteHeader();
    BOOL    WriteBody();
    BOOL    ReadEntireSensor(char* pLine, int iNumRanges);
    BOOL    ReadRules(void);
    BOOL    ReadEntireRule(char* pLine, char* pSensorName, int
iSensorIndex);
    int     FindSensor(char* pSensorName);
    BOOL    IsEmptyLine(char* pLine);
    char*   ReadNextLine(char* pLine, int iLength);
```

```
    void    AddTabs(int count);

    FILE*   m_pInputFile;
    FILE*   m_pOutputFile;
    char*   m_pOutFileName;
    BOOL    m_bWantBeeper;
    BOOL    m_bWantMotor;
    BOOL    m_bIsDigitalSensor;
    BOOL    m_bIsForwardMotor;
    BOOL    m_bIsBackwardMotor;
    int       m_iTabCount;

    int       m_iNumRules;
    int       m_iNumSensors;
    Sensor  m_pSensor[MAX_NUM_SENSORS];
    Rule    m_pRule[MAX_NUM_RULES];
    int     m_pOrder[MAX_NUM_RULES];
};
#endif /*_SENSORS_H*/
```

/* This is the sensor.cpp called in the main program

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#include "sensor.h"


/*
   Triangular
   Trapezoidal
*/
Sensors::Sensors()
   : m_pInputFile(NULL)
   , m_pOutputFile(NULL)
   , m_pOutFileName(NULL)
   , m_bWantBeeper(FALSE)
   , m_bWantMotor(FALSE)
   , m_bIsDigitalSensor(FALSE)
   , m_bIsForwardMotor(FALSE)
   , m_bIsBackwardMotor(FALSE)
   , m_iNumSensors(0)
   , m_iNumRules(0)
   , m_iTabCount(0)
{
```

```
    for (int i=0; i < MAX_NUM_RULES; i++)
    {
        m_pOrder[i] = 0;
    }
}

Sensors::~Sensors()
{
    if (m_pInputFile)
    {
        ::fclose(m_pInputFile);
        m_pInputFile  = 0;
    }

    if (m_pOutputFile)
    {
        ::fclose(m_pOutputFile);
        m_pOutputFile          = 0;
    }

    if (m_pOutFileName)
    {
        delete [] m_pOutFileName;
        m_pOutFileName       = 0;
    }
}

BOOL
Sensors::ReadInput(int argc, char* argv[])
{
    BOOL retVal = TRUE;
    if (argc < 3)
    {
        printf("\tUsage: %s <infile> <outfile>\n\n", argv[0]);
        return FALSE;
    }

    m_pInputFile = fopen(argv[1], "r");
    if (m_pInputFile == NULL)
    {
        printf("Error opening input file %s\n", argv[1]);
        return FALSE;
    }

    m_pOutFileName  = new char[strlen(argv[2]) + 1];
    strcpy(m_pOutFileName, argv[2]);
```

```
   retVal = ParseInput();
   if (!retVal)
   {
         printf("Data in input file is invalid. Please check and try
again..:)\n");
   }

   if (retVal)
   {
         retVal = WriteHeader();
         }

   if (retVal)
   {
         retVal = WriteBody();
   }

   return retVal;
}

BOOL
Sensors::ParseInput()
{
   char    cLine[MAX_LENGTH];
   char    cSensorName[MAX_LENGTH];
   int       iLowLimit = 0;
   int       iHighLimit = 0;
   int       iFuzzyFuncType = 0;
   int       iNumRanges;
   char*   pRet = NULL;
   BOOL    bIsDone = FALSE;
   int       numSensor = 0;
   int       retValue = 0;

   pRet = ReadNextLine(cLine, MAX_LENGTH);
   if (pRet == NULL)
   {
         return FALSE;
   }

   retValue = sscanf(cLine, "%s", cSensorName);
   /* make sure we have a valid input */
   if (retValue != 1)
   {
         return FALSE;
```

```
   }

   while(!bIsDone)
   {
         /* Is it end of declaration section */
         if (strcmp(strlwr(cSensorName), "begin") == 0)
         {
            bIsDone = TRUE;
            break;
         }

         /* We assume all sensors are analog sensors and have
          * the following format:
          * SensorName FuzzyFunctionType NumRanges Ranges...
          *
          * SensorName - Temperature, Humidity, Beeper etc.
          * FuzzyFunctionType - Traingular, Trapeziodal etc.
          *   Currently suported: Traingular (1)
          * NumRanges - 3, 5 or 7
          * Ranges -
          *  For Triangular - Each range has 3 values -
          *                          2 Base Vertices and 1 Apex
          */

         /* Read properties of an analog sensor */

         retValue = sscanf(cLine,"%s %d %d", cSensorName,
&iFuzzyFuncType, &iNumRanges);

         /* make sure we have a valid input */
         if (retValue != 3)
         {
            return FALSE;
         }

         /* We currently support only Triangular, left tail and right tail
fuzzy functions */
         if (iFuzzyFuncType != 1 && iFuzzyFuncType != 2 &&
iFuzzyFuncType != 3)
         {
            return FALSE;
         }

         printf(" function types okay .\n");
         /* We currently support only 3/5/7 number of ranges */
         if (iNumRanges != 3 && iNumRanges != 5 && iNumRanges != 7)
```

```
        {
           return FALSE;
        }

        printf(" num ranges okay..\n");
        if (!ReadEntireSensor(cLine, iNumRanges))
        {
           return FALSE;
        }
        printf(" reading entire sensor okay ..\n");
        /* Read next line There better be soimething there
         * in the file. Any valid input file will have some
         * conditions which start after Begin.
         */
        pRet = ReadNextLine(cLine, MAX_LENGTH);

        if (pRet == NULL)
        {
           return FALSE;
        }
        printf(" reading next line okay \n");
        retValue = sscanf(cLine, "%s", cSensorName);
        /* make sure we have a valid input */
        if (retValue != 1)
        {
           return FALSE;
        }
   }

   /* Add a dummy sensor used for completion of rules */
   /* We do not suport more than MAX_NUM_SENSORS sensors */
   if (m_iNumSensors >= MAX_NUM_SENSORS)
   {
        return FALSE;
   }
        printf(" num sensors not greater than max \n");

   strcpy(m_pSensor[m_iNumSensors].m_pSensorName, "dummy");
   m_pSensor[m_iNumSensors].m_bIsDummy = TRUE;
   m_iNumSensors++;

   /* Now read the rules */

   return ReadRules();
}
```

```
BOOL
Sensors::WriteHeader()
{
   m_pOutputFile = fopen(m_pOutFileName,"w+");

   if (m_pOutputFile == NULL)
   {
        printf("Error creating output file %s\n", m_pOutFileName);
        return FALSE;
   }

   if (m_bWantMotor)
   {
        /* We may need to modify this if motor speed is to be taken
         * as input from the user
         */
        fprintf(m_pOutputFile, "\n\n/*direction: 0 - forward 1-
backward*/\n");
        fprintf(m_pOutputFile, "void fuzzymotor (int port, int speed, int
direction)\n{\n");
        fprintf(m_pOutputFile, "\tif (direction == 0) {\n");
        fprintf(m_pOutputFile, "\t\tif (speed < 0) \n");
        fprintf(m_pOutputFile, "\t\t\tspeed = -speed;\n");
        fprintf(m_pOutputFile, "\t}\n");
        fprintf(m_pOutputFile, "\telse {\n");
        fprintf(m_pOutputFile, "\t\tif (speed > 0) \n");
        fprintf(m_pOutputFile, "\t\t\tspeed = -speed;\n");
        fprintf(m_pOutputFile, "\t}\n\n");
        fprintf(m_pOutputFile, "\tmotor(port,speed);\n");
        fprintf(m_pOutputFile, "\tmotor(port,0);\n");
        fprintf(m_pOutputFile, "}\n\n");
   }

#if 0
   if (m_bIsForwardMotor)
   {
        /* We may need to modify this if motor speed is to be taken
         * as input from the user
         */
        fprintf(m_pOutputFile, "\n\n void motorf (void)\n{\n");
        fprintf(m_pOutputFile, "\tint m1, var1[4] ={-16,0,16,0}; \n");
        fprintf(m_pOutputFile, "\tint m2, var2[4] ={0,-16,0,16};\n");
        fprintf(m_pOutputFile, "\tint i;\n\n");
        fprintf(m_pOutputFile, "\tfor (i=0; i<4; i++)\n\t{\n");
        fprintf(m_pOutputFile, "\t\tm1 = var1[i];\n");
        fprintf(m_pOutputFile, "\t\tm2 = var2[i];\n");
```

```
        fprintf(m_pOutputFile, "\t\tmotor(1,m1);\n");
        fprintf(m_pOutputFile, "\t\tmotor(2,m2);\n");
        fprintf(m_pOutputFile, "\t\tmsleep(20);\n");
        fprintf(m_pOutputFile, "\t}\n");
        fprintf(m_pOutputFile, "}\n\n");
    }

    if (m_bIsBackwardMotor)
    {
        /* We may need to modify this if motor speed is to be taken
         * as input from the user
         */
        fprintf(m_pOutputFile, "\n\n void motorf (void)\n{\n");
        fprintf(m_pOutputFile, "\tint m1, var1[4] ={16,0,-16,0}; \n");
        fprintf(m_pOutputFile, "\tint m2, var2[4] ={0,-16,0,16};\n");
        fprintf(m_pOutputFile, "\tint i;\n\n");
        fprintf(m_pOutputFile, "\tfor (i=0; i<4; i++)\n\t{\n");
        fprintf(m_pOutputFile, "\t\tm1 = var1[i];\n");
        fprintf(m_pOutputFile, "\t\tm2 = var2[i];\n");
        fprintf(m_pOutputFile, "\t\tmotor(1,m1);\n");
        fprintf(m_pOutputFile, "\t\tmotor(2,m2);\n");
        fprintf(m_pOutputFile, "\t\tmsleep(20);\n");
        fprintf(m_pOutputFile, "\t}\n");
        fprintf(m_pOutputFile, "}\n\n");
    }
#endif /*0*/

    fprintf(m_pOutputFile, "\n\nint CalculateMembership(int x, int low, int
apex, int high, float* membership)\n{\n");
    fprintf(m_pOutputFile,"\tif (x >= low && x <= high)\n\t{\n");
    fprintf(m_pOutputFile,"\t\tif (x <= apex)\n");
    fprintf(m_pOutputFile,"\t\t\t*membership = (float) (((x - low) )/((apex -
low)));\n");
    fprintf(m_pOutputFile,"\t\telse\n");
    fprintf(m_pOutputFile,"\t\t\t*membership = (float) (((high - x) )/((high -
apex)));\n");
    fprintf(m_pOutputFile,"\t\treturn 1;\n");
    fprintf(m_pOutputFile,"\t}\n\telse\n\t{\n");
    fprintf(m_pOutputFile,"\t\treturn 0;\n");
    fprintf(m_pOutputFile,"\t}\n}\n\n");
```

Appendix B

Code for VB program

Code for main screen

```
VERSION 5.00

Begin VB.Form Form1
  BackColor       =   &H80000007&
  Caption         =   "Form1"
  ClientHeight    =   5760
  ClientLeft      =   60
  ClientTop       =   345
  ClientWidth     =   7020
  FillColor       =   &H00800080&
  LinkTopic       =   "Form1"
  ScaleHeight     =   5760
  ScaleWidth      =   7020
  StartUpPosition =   1  'CenterOwner
  WindowState     =   2  'Maximized
  Begin VB.Frame Frame1
    BackColor       =   &H00000000&
    ForeColor       =   &H00FFFFFF&
    Height          =   2175
    Left            =   1920
    TabIndex        =   2
    Top             =   1920
    Width           =   6975
    Begin VB.Label Label1
      Alignment       =   2  'Center
      BackColor       =   &H00000000&
      Caption         =   "Fuzzy Expert System for Handy Board "
      BeginProperty Font
        Name            =   "Arial"
        Size            =   24
        Charset         =   0
        Weight          =   700
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00FFFFFF&
      Height          =   1215
      Left            =   360
      TabIndex        =   3
      Top             =   480
      Width           =   6255
    End
  End
  Begin VB.CommandButton cmdexit
    Caption         =   "Exit"
    BeginProperty Font
```

```
      Name           =   "MS Sans Serif"
      Size           =   8.25
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   615
    Left           =   6480
    TabIndex       =   1
    Top            =   6000
    Width          =   2175
  End
  Begin VB.CommandButton cmdproceed
    BackColor      =   &H80000007&
    Caption        =   "Proceed"
    BeginProperty Font
      Name           =   "MS Sans Serif"
      Size           =   9.75
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   615
    Left           =   1920
    MaskColor      =   &H00404080&
    TabIndex       =   0
    Top            =   6000
    Width          =   2175
  End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Private Sub cmdexit_Click()
End
End Sub

Private Sub cmdproceed_Click()
Rem testprint2
Hide
Form4.Show
End Sub
```

```
Sub TestPrint()
Open "f:\test.txt" For Output As #1
Print #1, "// Specify all the I/O sensors here"

i = 1
Do While i < 4
Print #1, ; " "; i; " "; i * 2; " "
i = i + 1
Loop

i = 5
Do While i < 8
Print #1, i; " "; i * 2; " ";
i = i + 1
Loop
Close #1
End Sub

Sub testprint2()
Open "f:\test.txt" For Output As #1
Print #1, "// Specify all the I/O sensors here";

iIndex = 1
Do While iIndex <= 3
Print #1,
Print #1, "SensName"; " "; 1; " "; 3; " ";
iNumRanges = 1
Do While iNumRanges <= 3
Print #1, 0; " "; 1; " "; 2; " ";
iNumRanges = iNumRanges + 1
Loop
iIndex = iIndex + 1
Loop
Print #1,
Print #1, "// specify rules"
Print #1, "begin"
iRuleIndex = 1
Do While iRuleIndex <= 2
Print #1, "SensName"; " "; 0; " ";
Print #1, 3; " "; 1; " "; 0; " "; "0 "
iRuleIndex = iRuleIndex + 1
Loop
Print #1, "end"
Close #1
End Sub

Private Sub Form_Load()
 blah = FullName
 tempDirName = CurDir
 Print blah
Rem  Form5.Show
```

```
Rem RunProgram.Show
End Sub
```

Code for 1st Form

```
VERSION 5.00
Begin VB.Form Form4
  Caption        =   "Form4"
  ClientHeight   =   3195
  ClientLeft     =   165
  ClientTop      =   735
  ClientWidth    =   4680
  LinkTopic      =   "Form4"
  ScaleHeight    =   8310
  ScaleWidth     =   11880
  StartUpPosition =   3  'Windows Default
  WindowState    =   2  'Maximized
  Begin VB.CommandButton Command7
    Caption        =   "Exit"
    BeginProperty Font
      Name           =   "Arial"
      Size           =   8.25
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   495
    Left           =   4560
    TabIndex       =   4
    Top            =   4920
    Width          =   2175
  End
  Begin VB.CommandButton Command5
    Caption        =   "Download to a handy board"
    BeginProperty Font
      Name           =   "MS Sans Serif"
      Size           =   9.75
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   495
    Left           =   2640
    TabIndex       =   3
    Top            =   2880
    Width          =   6015
  End
  Begin VB.CommandButton Command3
```

```
  Caption         =   "View source code"
  BeginProperty Font
    Name            =   "MS Sans Serif"
    Size            =   9.75
    Charset         =   0
    Weight          =   700
    Underline       =   0   'False
    Italic          =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  Height          =   495
  Left            =   2640
  TabIndex        =   2
  Top             =   2200
  Width           =   6015
End
Begin VB.CommandButton Command2
  Caption         =   "Run C Program for generating downloadable program"
  BeginProperty Font
    Name            =   "MS Sans Serif"
    Size            =   9.75
    Charset         =   0
    Weight          =   700
    Underline       =   0   'False
    Italic          =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  Height          =   495
  Left            =   2640
  TabIndex        =   1
  Top             =   1520
  Width           =   6015
End
Begin VB.CommandButton Command1
  Caption         =   "Input to fuzzy forms"
  BeginProperty Font
    Name            =   "MS Sans Serif"
    Size            =   9.75
    Charset         =   0
    Weight          =   700
    Underline       =   0   'False
    Italic          =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  Height          =   495
  Left            =   2640
  TabIndex        =   0
  Top             =   840
  Width           =   6015
End
Begin VB.Menu mnufile
```

```
    Caption         =   "&File"
  End
  Begin VB.Menu mnuhelp
    Caption         =   "&Help"
  End
End
Attribute VB_Name = "Form4"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Command1_Click()
Hide
Form2.Show
End Sub

Private Sub Command2_Click()
RunProgram.Show
End Sub

Private Sub Command3_Click()
Dim PathName As String
 PathName = "notepad.exe"
PathName = PathName + " "
PathName = PathName + RunProgram.OutFile.Text
Shell (PathName)
End Sub

Private Sub Command4_Click()
Dim PathName As String
PathName = "cl.exe"
PathName = PathName + " "
PathName = PathName + RunProgram.OutFile.Text
Shell (PathName)
End Sub

Private Sub Command5_Click()
Dim PathName As String
PathName = "command.com"
PathName = PathName + " "
PathName = PathName + RunProgram.OutFile.Text
Shell (PathName)
End Sub

Private Sub Command7_Click()
End
End Sub
```

Code for sensor info form

```
VERSION 5.00
Begin VB.Form Form2
  BackColor       =   &H80000002&
  Caption         =   "Information about sensors "
  ClientHeight    =   7605
  ClientLeft      =   60
  ClientTop       =   345
  ClientWidth     =   9945
  FillColor       =   &H00FFFFFF&
  LinkTopic       =   "Form2"
  ScaleHeight     =   7605
  ScaleWidth      =   9945
  StartUpPosition =   3  'Windows Default
  WindowState     =   2  'Maximized
  Begin VB.CommandButton Command1
    Caption         =   "Back"
    BeginProperty Font
      Name            =   "Arial"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Height          =   495
    Left            =   4200
    TabIndex        =   44
    Top             =   5400
    Width           =   1935
  End
  Begin VB.TextBox Text8
    Height          =   375
    Left            =   4320
    TabIndex        =   42
    Top             =   2640
    Width           =   1335
  End
  Begin VB.TextBox Text9
    Height          =   375
    Left            =   6120
    TabIndex        =   41
    Top             =   2640
    Width           =   1335
  End
  Begin VB.TextBox Text11
    Height          =   375
    Left            =   4320
    TabIndex        =   40
    Top             =   3120
```

```
  Width          =   1335
End
Begin VB.TextBox Text14
  Height         =   375
  Left           =   4320
  TabIndex       =   39
  Top            =   3600
  Width          =   1335
End
Begin VB.TextBox Text15
  Height         =   375
  Left           =   6120
  TabIndex       =   38
  Top            =   3600
  Width          =   1335
End
Begin VB.TextBox Text17
  Height         =   375
  Left           =   4320
  TabIndex       =   37
  Top            =   4080
  Width          =   1335
End
Begin VB.TextBox Text18
  Height         =   375
  Left           =   6120
  TabIndex       =   36
  Top            =   4080
  Width          =   1335
End
Begin VB.TextBox Text3
  Height         =   375
  Left           =   6120
  TabIndex       =   35
  Top            =   1680
  Width          =   1335
End
Begin VB.TextBox Text5
  Height         =   375
  Left           =   4320
  TabIndex       =   34
  Top            =   2160
  Width          =   1335
End
Begin VB.TextBox Text6
  Height         =   375
  Left           =   6120
  TabIndex       =   33
  Top            =   2160
  Width          =   1335
End
```

```
Begin VB.TextBox Text20
  Height        =   375
  Left          =   4320
  TabIndex      =   32
  Top           =   4560
  Width         =   1335
End
Begin VB.TextBox Text21
  Height        =   375
  Left          =   6120
  TabIndex      =   31
  Top           =   4560
  Width         =   1335
End
Begin VB.TextBox Text12
  Height        =   375
  Left          =   6120
  TabIndex      =   30
  Top           =   3120
  Width         =   1335
End
Begin VB.TextBox Text2
  Height        =   375
  Left          =   4320
  TabIndex      =   29
  Top           =   1680
  Width         =   1335
End
Begin VB.TextBox Text1
  Enabled       =   0   'False
  Height        =   375
  Left          =   2520
  TabIndex      =   27
  Top           =   1680
  Width         =   1335
End
Begin VB.CommandButton cmdadd
  Caption       =   "Add"
  BeginProperty Font
    Name          =   "Arial"
    Size          =   9.75
    Charset       =   0
    Weight        =   700
    Underline     =   0   'False
    Italic        =   0   'False
    Strikethrough =   0   'False
  EndProperty
  Height        =   495
  Left          =   1800
  TabIndex      =   23
  Top           =   5400
```

```
  Width          =   1575
End
Begin VB.TextBox Text19
  Enabled        =   0   'False
  Height         =   375
  Left           =   2520
  TabIndex       =   6
  Top            =   4560
  Width          =   1335
End
Begin VB.TextBox Text16
  Enabled        =   0   'False
  Height         =   375
  Left           =   2520
  TabIndex       =   5
  Top            =   4080
  Width          =   1335
End
Begin VB.TextBox Text7
  Enabled        =   0   'False
  Height         =   375
  Left           =   2520
  TabIndex       =   2
  Top            =   2640
  Width          =   1335
End
Begin VB.TextBox Text4
  Enabled        =   0   'False
  Height         =   375
  Left           =   2520
  TabIndex       =   1
  Top            =   2160
  Width          =   1335
End
Begin VB.TextBox Text13
  Enabled        =   0   'False
  Height         =   375
  Left           =   2520
  TabIndex       =   4
  Top            =   3600
  Width          =   1335
End
Begin VB.TextBox Text10
  Enabled        =   0   'False
  Height         =   375
  Left           =   2520
  TabIndex       =   3
  Top            =   3120
  Width          =   1335
End
Begin VB.ComboBox Combo1
```

```
  Enabled        =   0   'False
  BeginProperty Font
    Name           =   "Arial"
    Size           =   8.25
    Charset        =   0
    Weight         =   400
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   330
  Index          =   0
  ItemData       =   "sensors info2.frx":0000
  Left           =   3480
  List           =   "sensors info2.frx":000D
  TabIndex       =   15
  Text           =   "Fuzzy Type"
  Top            =   840
  Width          =   1455
End
Begin VB.CommandButton cmdexit
  Caption        =   "Exit"
  BeginProperty Font
    Name           =   "Arial"
    Size           =   9.75
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   495
  Left           =   7200
  TabIndex       =   13
  Top            =   5400
  Width          =   1575
End
Begin VB.CommandButton cmdruleform
  Caption        =   "Rule Form"
  BeginProperty Font
    Name           =   "Arial"
    Size           =   9.75
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   495
  Left           =   8040
  TabIndex       =   12
```

```
  Top           =   3960
  Width         =   1575
End
Begin VB.Frame Frame1
  Caption       =   "Options"
  BeginProperty Font
    Name          =   "Arial"
    Size          =   9.75
    Charset       =   0
    Weight        =   700
    Underline     =   0   'False
    Italic        =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  Height        =   1500
  Left          =   7920
  TabIndex      =   8
  Top           =   1320
  Width         =   1575
  Begin VB.OptionButton Option3
    Height        =   195
    Left          =   360
    TabIndex      =   26
    Top           =   1200
    Width         =   375
  End
  Begin VB.OptionButton Option2
    Height        =   195
    Left          =   360
    TabIndex      =   25
    Top           =   780
    Width         =   375
  End
  Begin VB.OptionButton Option1
    Height        =   195
    Left          =   360
    TabIndex      =   24
    Top           =   360
    Width         =   375
  End
  Begin VB.Label lbl7
    Alignment     =   2  'Center
    Appearance    =   0  'Flat
    BackColor     =   &H80000005&
    Caption       =   "7"
    BeginProperty Font
      Name          =   "MS Sans Serif"
      Size          =   12
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
```

```
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   300
    Left            =   960
    TabIndex        =   11
    Top             =   1080
    Width           =   300
  End
  Begin VB.Label lbl5
    Alignment       =   2  'Center
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    Caption         =   "5"
    BeginProperty Font
      Name          =   "MS Sans Serif"
      Size          =   12
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   300
    Left            =   960
    TabIndex        =   10
    Top             =   660
    Width           =   300
  End
  Begin VB.Label Label1
    Alignment       =   2  'Center
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    Caption         =   "3"
    BeginProperty Font
      Name          =   "MS Sans Serif"
      Size          =   12
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   300
    Left            =   960
    TabIndex        =   9
    Top             =   240
    Width           =   300
```

```
  End
End
Begin VB.ComboBox Combo3portno
  Enabled        =   0   'False
  BeginProperty Font
    Name           =   "Arial"
    Size           =   8.25
    Charset        =   0
    Weight         =   400
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   330
  Index          =   0
  ItemData       =   "sensors info2.frx":0034
  Left           =   5280
  List           =   "sensors info2.frx":004A
  TabIndex       =   7
  Text           =   "Port No."
  Top            =   840
  Width          =   1455
End
Begin VB.ComboBox Comboinputsensor
  BeginProperty Font
    Name           =   "Arial"
    Size           =   8.25
    Charset        =   0
    Weight         =   400
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   330
  Index          =   0
  ItemData       =   "sensors info2.frx":0060
  Left           =   1560
  List           =   "sensors info2.frx":0079
  TabIndex       =   0
  Text           =   "Sensor Type"
  Top            =   840
  Width          =   1455
End
Begin VB.Label Label2
  Caption        =   "Sensor Number: "
  Height         =   255
  Left           =   3600
  TabIndex       =   43
  Top            =   480
  Width          =   1455
End
```

```
Begin VB.Label SensorType
   Caption         =   "Sensor Type"
   Height          =   255
   Left            =   1680
   TabIndex        =   28
   Top             =   480
   Visible         =   0   'False
   Width           =   1095
End
Begin VB.Label lblvhigh
   BackColor       =   &H00800000&
   Caption         =   "V_High"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00FFFFFF&
   Height          =   375
   Left            =   1560
   TabIndex        =   22
   Top             =   4080
   Width           =   855
End
Begin VB.Label lblvvhigh
   BackColor       =   &H00800000&
   Caption         =   "V_V High"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   8.25
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00FFFFFF&
   Height          =   375
   Left            =   1560
   TabIndex        =   21
   Top             =   4560
   Width           =   855
End
Begin VB.Label lblvlow
   BackColor       =   &H00800000&
   Caption         =   "V_Low"
   BeginProperty Font
```

```
    Name           =   "MS Sans Serif"
    Size           =   8.25
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  ForeColor      =   &H00FFFFFF&
  Height         =   375
  Left           =   1560
  TabIndex       =   20
  Top            =   2160
  Width          =   855
End
Begin VB.Label lbllow
  BackColor      =   &H00800000&
  Caption        =   "Low"
  BeginProperty Font
    Name           =   "MS Sans Serif"
    Size           =   8.25
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  ForeColor      =   &H00FFFFFF&
  Height         =   375
  Left           =   1560
  TabIndex       =   19
  Top            =   2640
  Width          =   855
End
Begin VB.Label lblmedium
  BackColor      =   &H00800000&
  Caption        =   "Medium"
  BeginProperty Font
    Name           =   "MS Sans Serif"
    Size           =   8.25
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  ForeColor      =   &H00FFFFFF&
  Height         =   375
  Left           =   1560
  TabIndex       =   18
  Top            =   3120
```

```
  Width          =   855
End
Begin VB.Label lblhigh
  BackColor      =   &H00800000&
  Caption        =   "High"
  BeginProperty Font
    Name           =   "MS Sans Serif"
    Size           =   8.25
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  ForeColor      =   &H00FFFFFF&
  Height         =   375
  Left           =   1560
  TabIndex       =   17
  Top            =   3600
  Width          =   855
End
Begin VB.Label lblvvlow
  BackColor      =   &H00800000&
  Caption        =   "V_V Low"
  BeginProperty Font
    Name           =   "MS Sans Serif"
    Size           =   8.25
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  ForeColor      =   &H00FFFFFF&
  Height         =   375
  Left           =   1560
  TabIndex       =   16
  Top            =   1680
  Width          =   855
End
Begin VB.Label lblranges
  Alignment      =   2  'Center
  BackColor      =   &H00800000&
  Caption        =   "Ranges"
  BeginProperty Font
    Name           =   "Arial"
    Size           =   9.75
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
```

```
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H00FFFFFF&
    Height          =   375
    Left            =   7920
    TabIndex        =   14
    Top             =   840
    Width           =   1575
  End
End
Attribute VB_Name = "Form2"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Private Sub cmdadd_Click()
m_bisValid = False
ValidateInput
If m_bisValid = True Then

i = 0
Do While i < Comboinputsensor(0).ListCount()
If m_pTempSensorName = Comboinputsensor(0).List(i) Then
Comboinputsensor(0).RemoveItem (i)
Exit Do
End If
i = i + 1
Loop

i = 0
Do While i < Combo3portno(0).ListCount()
If m_uTempPortNumber = Val(Combo3portno(0).List(i)) Then
Combo3portno(0).RemoveItem (i)
Exit Do
End If
i = i + 1
Loop

CreateNewSensor
ResetFuzzyVals
End If
End Sub

Private Sub ValidateInput()
Initialize
m_bisValid = False
If NumSensors >= 7 Then
MsgBox "You have already reached a max limit of 7 sensors..."
ElseIf ValidSensorInput = False Then
```

```
MsgBox "Select a valid sensor..."
ElseIf ValidFuzzyInput = False Then
MsgBox "Select a valid fuzzification type..."
ElseIf ValidPortInput = False Then
MsgBox "Select a valid port..."
ElseIf ValidOptionInput = False Then
MsgBox "Select a valid option for number of ranges..."
Else
m_bisValid = True
If m_uTempNumRanges = 3 Then
ValidateRange3
ElseIf m_uTempNumRanges = 5 Then
ValidateRange5
ElseIf m_uTempNumRanges = 7 Then
ValidateRange7
Else
m_bisValid = False
End If
End If
End Sub

Private Sub cmdback_Click()
Unload Form2
End Sub

Private Sub cmdexit_Click()
End
End Sub

Private Sub cmdruleform_Click()
If NumInputSensors = 0 Then
MsgBox "Please input atleast one input sensor..."
Exit Sub
End If
If NumOutputSensors = 0 Then
MsgBox "Please input atleast one output sensor..."
Exit Sub
End If
Hide
Form3.Show
End Sub

Private Sub Combo1_Click(IndexIn As Integer)
Initialize
If Combo1(IndexIn) = "" Then
ValdFuzzyInput = False
MsgBox "U have not selected from the list "
Else
ValidFuzzyInput = True
If Combo1(IndexIn) = "Triangular" Then
m_uTempFuzzyType = 1
```

```
ElseIf Combo1(IndexIn) = "Left Tail" Then
m_uTempFuzzyType = 2
Else
m_uTempFuzzyType = 3
End If
End If
End Sub

Private Sub Combo3portno_Change(IndexIn As Integer)
Initialize
Combo3portno_Click (IndexIn)
End Sub

Private Sub Combo3portno_Click(IndexIn As Integer)
If Combo3portno(IndexIn) = "" Then
ValidPortInput = False
MsgBox "U have not selected from the list "
Else
ValidPortInput = True
m_uTempPortNumber = Val(Combo3portno(IndexIn))
End If
End Sub

Private Sub Comboinputsensor_Change(Index As Integer)
Comboinputsensor_Click (Index)
End Sub


Private Sub Comboinputsensor_Click(IndexIn As Integer)
Initialize
If Comboinputsensor(IndexIn) = "" Then
ValidSensorInput = False
SensorType.Visible = False
Combo1(0).Enabled = False
Combo3portno(0).Enabled = False
MsgBox "U have not selected from the list "
Else
ValidSensorInput = True
m_pTempSensorName = Comboinputsensor(IndexIn)
m_pTempSensorName = Trim(m_pTempSensorName)
SensorType.Visible = True
Combo1(0).Enabled = True
Combo3portno(0).Enabled = True
PortRequired = True
If Comboinputsensor(IndexIn) = "Potentiometer1" Then
SensorType.Caption = "Input"
m_bTempInpuSensor = True
ElseIf Comboinputsensor(IndexIn) = "Potentiometer2" Then
SensorType.Caption = "Input"
m_bTempInpuSensor = True
ElseIf Comboinputsensor(IndexIn) = "IR_Sensor" Then
```

```
SensorType.Caption = "Input"
m_bTempInpuSensor = True
ElseIf Comboinputsensor(IndexIn) = "Temperature" Then
SensorType.Caption = "Input"
m_bTempInpuSensor = True
ElseIf Comboinputsensor(IndexIn) = "Motor1" Then
SensorType.Caption = "Output"
m_bTempInpuSensor = False
ElseIf Comboinputsensor(IndexIn) = "Motor2" Then
SensorType.Caption = "Output"
m_bTempInpuSensor = False
ElseIf Comboinputsensor(IndexIn) = "Motor3" Then
SensorType.Caption = "Output"
m_bTempInpuSensor = False
End If
End If
End Sub


Private Sub Command1_Click()
Hide
Form4.Show
End Sub

Private Sub Form_Load()
Initialize
Label2.Caption = "Sensor Number: " + Str(NumSensors + 1)
End Sub

Private Sub Option1_Click()
Initialize
ValidOptionInput = True
m_uTempNumRanges = 3
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
Text18.Text = ""
```

```
Text19.Text = ""
Text20.Text = ""
Text21.Text = ""
Text1.Enabled = False
Text2.Enabled = False
Text3.Enabled = False
Text4.Enabled = False
Text5.Enabled = False
Text6.Enabled = False
Text16.Enabled = False
Text17.Enabled = False
Text18.Enabled = False
Text19.Enabled = False
Text20.Enabled = False
Text21.Enabled = False
Text7.Enabled = True
Text8.Enabled = True
Text9.Enabled = True
Text10.Enabled = True
Text11.Enabled = True
Text12.Enabled = True
Text13.Enabled = True
Text14.Enabled = True
Text15.Enabled = True
End Sub


Private Sub option2_Click()
Initialize
ValidOptionInput = True
m_uTempNumRanges = 5
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
Text18.Text = ""
Text19.Text = ""
```

```
Text20.Text = ""
Text21.Text = ""
Text1.Enabled = False
Text2.Enabled = False
Text3.Enabled = False
Text19.Enabled = False
Text20.Enabled = False
Text21.Enabled = False
Text4.Enabled = True
Text5.Enabled = True
Text6.Enabled = True
Text7.Enabled = True
Text8.Enabled = True
Text9.Enabled = True
Text10.Enabled = True
Text11.Enabled = True
Text12.Enabled = True
Text13.Enabled = True
Text14.Enabled = True
Text15.Enabled = True
Text16.Enabled = True
Text17.Enabled = True
Text18.Enabled = True
End Sub


Private Sub option3_Click()
Initialize
ValidOptionInput = True
m_uTempNumRanges = 7
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
Text18.Text = ""
Text19.Text = ""
Text20.Text = ""
Text21.Text = ""
```

```
Text1.Enabled = True
Text2.Enabled = True
Text3.Enabled = True
Text4.Enabled = True
Text5.Enabled = True
Text6.Enabled = True
Text7.Enabled = True
Text8.Enabled = True
Text9.Enabled = True
Text10.Enabled = True
Text11.Enabled = True
Text12.Enabled = True
Text13.Enabled = True
Text14.Enabled = True
Text15.Enabled = True
Text16.Enabled = True
Text17.Enabled = True
Text18.Enabled = True
Text19.Enabled = True
Text20.Enabled = True
Text21.Enabled = True
End Sub

Private Sub Text1_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text1.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text1.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text1.Text = Val(Text1.Text)
End If
End Sub



Private Sub Text10_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text10.Text) < 0 Then
MsgBox " Value less than 0 can not be entered"
ElseIf Val(Text10.Text) > 255 Then
MsgBox " Val greater than 255 cannot be entered"
Else
Text10.Text = Val(Text10.Text)
End If
End Sub

Private Sub Text11_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text11.Text) <= 0 Then
```

```
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text11.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text11.Text = Val(Text11.Text)
End If
End Sub

Private Sub Text12_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text12.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text12.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text12.Text = Val(Text12.Text)
End If
End Sub

Private Sub Text13_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text13.Text) < 0 Then
MsgBox " Value less than zero cannot be entered"
ElseIf Val(Text13.Text) > 255 Then
MsgBox " Val greater than zero cannot be entered"
ElseIf Val(Text13.Text) = 0 Then
MsgBox "Enter some value"
Else
End If
End Sub

Private Sub Text14_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text14.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text14.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text14.Text = Val(Text14.Text)
End If
End Sub

Private Sub Text15_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text15.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text15.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text15.Text = Val(Text15.Text)
End If
```

```
End Sub

Private Sub Text16_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text16.Text) < 0 Then
MsgBox " Value less than 0 can not be entered"
ElseIf Val(Text16.Text) > 255 Then
MsgBox " Val greater than 255 cannot be entered"
Else
Text16.Text = Val(Text16.Text)
End If
End Sub

Private Sub Text17_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text17.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text17.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text17.Text = Val(Text17.Text)
End If
End Sub

Private Sub Text18_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text18.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text18.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text18.Text = Val(Text18.Text)
End If
End Sub

Private Sub Text19_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text19.Text) < 0 Then
MsgBox " Value less than 0 can not be entered"
ElseIf Val(Text19.Text) > 255 Then
MsgBox " Val greater than 255 cannot be entered"
Else
Text19.Text = Val(Text19.Text)
End If
End Sub

Private Sub Text2_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text2.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text2.Text) > 255 Then
```

```
MsgBox " Value more than 255 cannot be entered"
Else
Text2.Text = Val(Text2.Text)
End If
End Sub

Private Sub Text20_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text20.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text20.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text20.Text = Val(Text20.Text)
End If
End Sub

Private Sub Text21_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text21.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text21.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text21.Text = Val(Text21.Text)
End If
End Sub

Private Sub Text3_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text3.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text3.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text3.Text = Val(Text3.Text)
End If
End Sub

Private Sub Text4_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text4.Text) < 0 Then
MsgBox " Value less than 0 can not be entered"
ElseIf Val(Text4.Text) > 255 Then
MsgBox " Val greater than 255 cannot be entered"
Else
Text4.Text = Val(Text4.Text)
End If
End Sub

Private Sub Text5_Change()
```

```
If m_bInResetMode = True Then Exit Sub
If Val(Text5.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text5.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text5.Text = Val(Text5.Text)
End If
End Sub

Private Sub Text6_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text6.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text6.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text6.Text = Val(Text6.Text)
End If
End Sub

Private Sub Text7_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text7.Text) < 0 Then
MsgBox " Value less than 0 can not be entered"
ElseIf Val(Text7.Text) > 255 Then
MsgBox " Val greater than 255 cannot be entered"
Else
Text7.Text = Val(Text7.Text)
End If
End Sub

Private Sub Text8_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text8.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text8.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text8.Text = Val(Text8.Text)
End If
End Sub

Private Sub Text9_Change()
If m_bInResetMode = True Then Exit Sub
If Val(Text9.Text) <= 0 Then
MsgBox " Value less than or equal to zero can not be entered"
ElseIf Val(Text9.Text) > 255 Then
MsgBox " Value more than 255 cannot be entered"
Else
Text9.Text = Val(Text9.Text)
```

```
End If
End Sub


Public Sub ResetFuzzyVals()
ResetValues
m_bInResetMode = True
Label2.Caption = "Sensor Number: " + Str(NumSensors + 1)
SensorType.Visible = False
Combo1(0).Enabled = False
Combo3portno(0).Enabled = False

Comboinputsensor(0).Text = "Sensor Type"
Combo1(0).Text = "Fuzzy Type"
Combo3portno(0).Text = "Port No."

Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
Text18.Text = ""
Text19.Text = ""
Text20.Text = ""
Text21.Text = ""
Text1.Enabled = False
Text2.Enabled = False
Text3.Enabled = False
Text4.Enabled = False
Text5.Enabled = False
Text6.Enabled = False
Text7.Enabled = False
Text8.Enabled = False
Text9.Enabled = False
Text10.Enabled = False
Text11.Enabled = False
Text12.Enabled = False
Text13.Enabled = False
Text14.Enabled = False
```

```
Text15.Enabled = False
Text16.Enabled = False
Text17.Enabled = False
Text18.Enabled = False
Text19.Enabled = False
Text20.Enabled = False
Text21.Enabled = False

Option1.Value = False
Option2.Value = False
Option3.Value = False
m_bInResetMode = False
End Sub
```

Code for Rule Form

```
VERSION 5.00
Begin VB.Form Form3
   BackColor       =   &H00C00000&
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Rule Form"
   ClientHeight    =   6570
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   10155
   LinkTopic       =   "Form3"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   6570
   ScaleWidth      =   10155
   StartUpPosition =   3  'Windows Default
   WindowState     =   2  'Maximized
   Begin VB.ComboBox Direction
      Height          =   315
      Index           =   3
      ItemData        =   "Rule form.frx":0000
      Left            =   6240
      List            =   "Rule form.frx":000A
      TabIndex        =   26
      Text            =   "Direction"
      Top             =   3960
      Width           =   1575
   End
   Begin VB.ComboBox Direction
      Height          =   315
      Index           =   2
      ItemData        =   "Rule form.frx":0021
      Left            =   6240
      List            =   "Rule form.frx":002B
      TabIndex        =   25
      Text            =   "Direction"
      Top             =   3240
```

```
   Width          =   1575
End
Begin VB.ComboBox Direction
   Height         =   315
   Index          =   1
   ItemData       =   "Rule form.frx":0042
   Left           =   6240
   List           =   "Rule form.frx":004C
   TabIndex       =   24
   Text           =   "Direction"
   Top            =   2520
   Width          =   1575
End
Begin VB.CommandButton Command1
   Caption        =   "Back To Main Screen"
   BeginProperty Font
      Name           =   "Arial"
      Size           =   9.75
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
   EndProperty
   Height         =   375
   Left           =   3480
   TabIndex       =   23
   Top            =   5400
   Width          =   2175
End
Begin VB.ComboBox SensorVal2
   Height         =   315
   ItemData       =   "Rule form.frx":0063
   Left           =   3960
   List           =   "Rule form.frx":007C
   TabIndex       =   22
   Text           =   "select"
   Top            =   1560
   Width          =   1575
End
Begin VB.ComboBox InCombo2
   Height         =   315
   ItemData       =   "Rule form.frx":00B5
   Left           =   1320
   List           =   "Rule form.frx":00C2
   TabIndex       =   21
   Text           =   "select"
   Top            =   1560
   Width          =   1575
End
Begin VB.ComboBox ComboAndOR
```

```
   Height        =   315
   ItemData      =   "Rule form.frx":00F9
   Left          =   6360
   List          =   "Rule form.frx":0106
   TabIndex      =   20
   Text          =   "select"
   Top           =   960
   Width         =   1575
End
Begin VB.ComboBox SensorVal1
   Height        =   315
   ItemData      =   "Rule form.frx":0119
   Left          =   3960
   List          =   "Rule form.frx":0132
   TabIndex      =   19
   Text          =   "select"
   Top           =   960
   Width         =   1575
End
Begin VB.ComboBox InCombo1
   Height        =   315
   ItemData      =   "Rule form.frx":016B
   Left          =   1320
   List          =   "Rule form.frx":0178
   TabIndex      =   18
   Text          =   "select"
   Top           =   960
   Width         =   1575
End
Begin VB.ComboBox OutValCombo
   Height        =   315
   Index         =   3
   ItemData      =   "Rule form.frx":01AF
   Left          =   3840
   List          =   "Rule form.frx":01C8
   TabIndex      =   17
   Text          =   "select"
   Top           =   3960
   Width         =   1575
End
Begin VB.ComboBox OutValCombo
   Height        =   315
   Index         =   2
   ItemData      =   "Rule form.frx":0201
   Left          =   3840
   List          =   "Rule form.frx":021A
   TabIndex      =   16
   Text          =   "select"
   Top           =   3240
   Width         =   1575
End
```

```
Begin VB.ComboBox OutValCombo
   Height        =   315
   Index         =   1
   ItemData      =   "Rule form.frx":0253
   Left          =   3840
   List          =   "Rule form.frx":026C
   TabIndex      =   15
   Text          =   "select"
   Top           =   2520
   Width         =   1575
End
Begin VB.ComboBox OutCombo
   Height        =   315
   Index         =   3
   ItemData      =   "Rule form.frx":02A5
   Left          =   1320
   List          =   "Rule form.frx":02B2
   TabIndex      =   5
   Text          =   "select"
   Top           =   3960
   Width         =   1575
End
Begin VB.ComboBox OutCombo
   Height        =   315
   Index         =   2
   ItemData      =   "Rule form.frx":02D1
   Left          =   1320
   List          =   "Rule form.frx":02DE
   TabIndex      =   3
   Text          =   "select"
   Top           =   3240
   Width         =   1575
End
Begin VB.ComboBox OutCombo
   Height        =   315
   Index         =   1
   ItemData      =   "Rule form.frx":02FD
   Left          =   1320
   List          =   "Rule form.frx":030A
   TabIndex      =   1
   Text          =   "select"
   Top           =   2520
   Width         =   1575
End
Begin VB.CommandButton cmdadd
   Caption       =   "Add"
   BeginProperty Font
      Name          =   "Arial"
      Size          =   9.75
      Charset       =   0
      Weight        =   700
```

```
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   375
  Left           =   1080
  MaskColor      =   &H00FFFFFF&
  TabIndex       =   14
  Top            =   5400
  Width          =   1455
End
Begin VB.CommandButton cmdsave
  Caption        =   "Save"
  BeginProperty Font
    Name           =   "Arial"
    Size           =   9.75
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   375
  Left           =   6120
  TabIndex       =   9
  Top            =   4560
  Width          =   1455
End
Begin VB.CommandButton cmdexit
  Caption        =   "Exit"
  BeginProperty Font
    Name           =   "Arial"
    Size           =   9.75
    Charset        =   0
    Weight         =   700
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   375
  Left           =   6120
  TabIndex       =   4
  Top            =   5400
  Width          =   1455
End
Begin VB.Label Label4
  BackColor      =   &H00C00000&
  Caption        =   "Is"
  DataMember     =   "Form2.Show"
  BeginProperty Font
    Name           =   "Arial"
```

```
      Size           =   8.25
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H00FFFFFF&
    Height         =   375
    Left           =   3240
    TabIndex       =   13
    Top            =   3300
    Width          =   255
  End
  Begin VB.Label Label3
    BackColor      =   &H00C00000&
    Caption        =   "Is"
    DataMember     =   "Form2.Show"
    BeginProperty Font
      Name           =   "Arial"
      Size           =   8.25
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H00FFFFFF&
    Height         =   375
    Left           =   3240
    TabIndex       =   12
    Top            =   3960
    Width          =   255
  End
  Begin VB.Label Label1
    BackColor      =   &H00C00000&
    Caption        =   "Is"
    DataMember     =   "Form2.Show"
    BeginProperty Font
      Name           =   "Arial"
      Size           =   8.25
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H00FFFFFF&
    Height         =   375
    Left           =   3240
    TabIndex       =   11
```

```
  Top             =   2520
  Width           =   255
End
Begin VB.Label lblthen
  BackColor       =   &H00C00000&
  Caption         =   "Then"
  BeginProperty Font
    Name          =   "Arial"
    Size          =   8.25
    Charset       =   0
    Weight        =   700
    Underline     =   0   'False
    Italic        =   0   'False
    Strikethrough =   0   'False
  EndProperty
  ForeColor       =   &H00FFFFFF&
  Height          =   255
  Left            =   360
  TabIndex        =   10
  Top             =   3000
  Width           =   495
End
Begin VB.Label Label2
  BackColor       =   &H00C00000&
  Caption         =   "Is"
  BeginProperty Font
    Name          =   "MS Sans Serif"
    Size          =   9.75
    Charset       =   0
    Weight        =   700
    Underline     =   0   'False
    Italic        =   0   'False
    Strikethrough =   0   'False
  EndProperty
  ForeColor       =   &H00FFFFFF&
  Height          =   255
  Left            =   3240
  TabIndex        =   8
  Top             =   1560
  Width           =   375
End
Begin VB.Label lblis
  BackColor       =   &H00C00000&
  Caption         =   "Is"
  BeginProperty Font
    Name          =   "MS Sans Serif"
    Size          =   9.75
    Charset       =   0
    Weight        =   700
    Underline     =   0   'False
    Italic        =   0   'False
```

```
    Strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H00FFFFFF&
  Height          =   255
  Left            =   3240
  TabIndex        =   7
  Top             =   960
  Width           =   375
End
Begin VB.Label lblif
  BackColor       =   &H00C00000&
  Caption         =   "If"
  BeginProperty Font
    Name            =   "Arial"
    Size            =   9.75
    Charset         =   0
    Weight          =   700
    Underline       =   0   'False
    Italic          =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H00FFFFFF&
  Height          =   255
  Left            =   480
  TabIndex        =   6
  Top             =   960
  Width           =   495
End
Begin VB.Label lblinput
  Alignment       =   2  'Center
  BackColor       =   &H00C00000&
  Caption         =   "Input"
  BeginProperty Font
    Name            =   "Arial"
    Size            =   12
    Charset         =   0
    Weight          =   700
    Underline       =   0   'False
    Italic          =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H00FFFFFF&
  Height          =   615
  Left            =   0
  TabIndex        =   2
  Top             =   120
  Width           =   1515
End
Begin VB.Label lbloutput
  Alignment       =   2  'Center
  BackColor       =   &H00C00000&
```

```
    Caption         =   "Output"
    BeginProperty Font
      Name          =   "Arial"
      Size          =   12
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H00FFFFFF&
    Height          =   375
    Left            =   0
    TabIndex        =   0
    Top             =   2040
    Width           =   1275
  End
End
Attribute VB_Name = "Form3"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub cmdadd_Click()
m_bisValid = False
ValidateRuleInput
If m_bisValid = True Then
CreateNewRule
Rem we were able to add a valid rule
Rem reset all variables for the next rule
InitializeRuleForm
End If
End Sub


Sub ValidateRuleInput()
RuleInitialize
m_bisValid = False
If NumRules >= 35 Then
MsgBox "You have already reached a max limit of 7 rules..."
Exit Sub
End If
If m_uNumInSensor = 0 Then
MsgBox "Select atleast one input sensor..."
Exit Sub
End If
If m_uNumOutSensor = 0 Then
MsgBox "Select atleast one output sensor..."
Exit Sub
End If
```

```
If Not TempInputRule1.m_iCondition = 0 Then
If m_uNumInSensor = 1 Then
MsgBox "Select the second input sensor for conditional clause..."
Exit Sub
End If
End If

If m_bCompleteRuleEntered = False Then
MsgBox "Select a valid value for the chosen sensor..."
Exit Sub
End If

m_bisValid = True
End Sub

Private Sub cmdsave_Click()
If NumRules = 0 Then
MsgBox "Please input atleast one rule..."
Exit Sub
End If

m_bToBeSaved = False
Form5.Show vbModal, Me
If m_bToBeSaved = False Then
Exit Sub
End If

Open Form5.InFile.Text For Output As #1
Print #1, "// Specify all the I/O sensors here";

iIndex = 1
Do While iIndex <= NumSensors
Print #1,
Print #1, m_pSensor(iIndex).m_pSensorName; " "; m_pSensor(iIndex).m_iFuzzyFuncType; " ";
m_pSensor(iIndex).m_iNumRanges; " ";
iNumRanges = 1
Do While iNumRanges <= m_pSensor(iIndex).m_iNumRanges
Print #1, m_pSensor(iIndex).m_pLowRange(iNumRanges); " ";
m_pSensor(iIndex).m_pApex(iNumRanges); " ";
m_pSensor(iIndex).m_pHighRange(iNumRanges); " ";
iNumRanges = iNumRanges + 1
Loop
iIndex = iIndex + 1
Loop
Print #1,
Print #1, "// specify rules"
Print #1, "begin"
iRuleIndex = 1
Do While iRuleIndex <= NumRules
iSensorIndex = m_pRule(iRuleIndex).m_iSensorIndex
Rem check if it is a dummy rule
```

```
If iSensorIndex = -1 Then
Print #1, "dummy "; " "; m_pRule(iRuleIndex).m_iOrder
Else
Print #1, m_pSensor(iSensorIndex).m_pSensorName; " "; m_pRule(iRuleIndex).m_iOrder; " ";
If m_pSensor(iSensorIndex).m_bInputSensor = True Then
Print #1, m_pSensor(iSensorIndex).m_iPort; " "; m_pRule(iRuleIndex).m_iState; " ";
m_pRule(iRuleIndex).m_iCondition; " "; "0 "
Else
Print #1, m_pSensor(iSensorIndex).m_iPort; " "; m_pRule(iRuleIndex).m_iState; " ";
m_pRule(iRuleIndex).m_iDirection
End If
End If
iRuleIndex = iRuleIndex + 1
Loop
Print #1, "end"
Close #1
End Sub

Private Sub Command1_Click()
Hide
Form4.Show
End Sub


Private Sub Form_Load()
InitializeRuleForm
End Sub

Private Sub InitializeRuleForm()
m_bInResetMode = True
InCombo1.Enabled = True
InCombo1.Text = "Select"

InCombo2.Enabled = True
InCombo2.Text = "Select"
InCombo2.Enabled = False
SensorVal1.Enabled = True
SensorVal2.Enabled = True
SensorVal1.Text = "Select"
SensorVal2.Text = "Select"
SensorVal1.Enabled = False
SensorVal2.Enabled = False
i = 1
Do While i <= 3
OutCombo(i).Enabled = True
OutValCombo(i).Enabled = True
Direction(i).Enabled = True
OutCombo(i).Text = "Select"
OutValCombo(i).Text = "Select"
Direction(i).Text = "Direction"
OutCombo(i).Enabled = False
```

```
OutValCombo(i).Enabled = False
Direction(i).Enabled = False
i = i + 1
Loop
ComboAndOR.Enabled = True
ComboAndOR.Text = "Select"
ComboAndOR.Enabled = False
ResetRuleVals
m_bInResetMode = False
End Sub



Private Sub cmdexit_Click()
End
End Sub

Private Sub InCombo1_DropDown()
Do While InCombo1.ListCount() > 0
InCombo1.RemoveItem (0)
Loop
iTempNumSensor = 1
Do While iTempNumSensor <= NumSensors
If m_pSensor(iTempNumSensor).m_bInputSensor = True Then
InCombo1.AddItem (m_pSensor(iTempNumSensor).m_pSensorName)
End If
iTempNumSensor = iTempNumSensor + 1
Loop
End Sub

Private Sub InCombo1_Change()
InCombo1_Click
End Sub

Private Sub InCombo1_Click()
If m_bInResetMode = True Then
Exit Sub
End If

RuleInitialize
If InCombo1 = "" Then
MsgBox "U have not selected from the list "
InCombo1.Text = "Select"
Exit Sub
End If
m_uNumInSensor = 1
m_bCompleteRuleEntered = False
SensorVal1.Enabled = True

Do While SensorVal1.ListCount() > 0
SensorVal1.RemoveItem (0)
```

```
Loop

iTempNumSensor = 1
Do While iTempNumSensor <= NumSensors
If InCombo1 = m_pSensor(iTempNumSensor).m_pSensorName Then
TempInputRule1.m_iSensorIndex = iTempNumSensor
If m_pSensor(iTempNumSensor).m_iNumRanges = 3 Then
SensorVal1.AddItem ("High")
SensorVal1.AddItem ("Medium")
SensorVal1.AddItem ("Low")
ElseIf m_pSensor(iTempNumSensor).m_iNumRanges = 5 Then
SensorVal1.AddItem ("V_High")
SensorVal1.AddItem ("High")
SensorVal1.AddItem ("Medium")
SensorVal1.AddItem ("Low")
SensorVal1.AddItem ("V_Low")
Else
SensorVal1.AddItem ("V_V High")
SensorVal1.AddItem ("V_High")
SensorVal1.AddItem ("High")
SensorVal1.AddItem ("Medium")
SensorVal1.AddItem ("Low")
SensorVal1.AddItem ("V_Low")
SensorVal1.AddItem ("V_V Low")
End If
Exit Do
End If
iTempNumSensor = iTempNumSensor + 1
Loop
End Sub


Private Sub InCombo2_DropDown()
Do While InCombo2.ListCount() > 0
InCombo2.RemoveItem (0)
Loop
iTempNumSensor = 1
Do While iTempNumSensor <= NumSensors
If m_pSensor(iTempNumSensor).m_bInputSensor = True Then
If Not InCombo1 = m_pSensor(iTempNumSensor).m_pSensorName Then
InCombo2.AddItem (m_pSensor(iTempNumSensor).m_pSensorName)
End If
End If
iTempNumSensor = iTempNumSensor + 1
Loop
End Sub

Private Sub InCombo2_Change()
InCombo2_Click
End Sub
```

```
Private Sub InCombo2_Click()
If m_bInResetMode = True Then
Exit Sub
End If
RuleInitialize
If InCombo2 = "" Then
MsgBox "U have not selected from the list "
InCombo2.Text = "Select"
Exit Sub
End If
m_uNumInSensor = 2
m_bCompleteRuleEntered = False
SensorVal2.Enabled = True

Do While SensorVal2.ListCount() > 0
SensorVal2.RemoveItem (0)
Loop

iTempNumSensor = 1
Do While iTempNumSensor <= NumSensors
If InCombo2 = m_pSensor(iTempNumSensor).m_pSensorName Then
TempInputRule2.m_iSensorIndex = iTempNumSensor
If m_pSensor(iTempNumSensor).m_iNumRanges = 3 Then
SensorVal2.AddItem ("High")
SensorVal2.AddItem ("Medium")
SensorVal2.AddItem ("Low")
ElseIf m_pSensor(iTempNumSensor).m_iNumRanges = 5 Then
SensorVal2.AddItem ("V_High")
SensorVal2.AddItem ("High")
SensorVal2.AddItem ("Medium")
SensorVal2.AddItem ("Low")
SensorVal2.AddItem ("V_Low")
Else
SensorVal2.AddItem ("V_V High")
SensorVal2.AddItem ("V_High")
SensorVal2.AddItem ("High")
SensorVal2.AddItem ("Medium")
SensorVal2.AddItem ("Low")
SensorVal2.AddItem ("V_Low")
SensorVal2.AddItem ("V_V Low")
End If
Exit Do
End If
iTempNumSensor = iTempNumSensor + 1
Loop
End Sub



Private Sub SensorVal1_Change()
SensorVal1_Click
```

```
End Sub

Private Sub SensorVal1_Click()
If m_bInResetMode = True Then
Exit Sub
End If
If SensorVal1 = "" Then
MsgBox "U have not selected from the list "
SensorVal1.Text = "Select"
Exit Sub
End If

If NumInputSensors > 1 Then
ComboAndOR.Enabled = True 'enable and/or sensor combo
End If

OutCombo(1).Enabled = True 'enable first output sensor combo

TempInputRule1.m_iCondition = 0
m_bCompleteRuleEntered = True

If SensorVal1 = "V_V High" Then
TempInputRule1.m_iState = 1
ElseIf SensorVal1 = "V_V Low" Then
TempInputRule1.m_iState = 7
ElseIf SensorVal1 = "V_High" Then
If m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule1.m_iState = 2
Else
TempInputRule1.m_iState = 1
End If
ElseIf SensorVal1 = "High" Then
If m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule1.m_iState = 3
ElseIf m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 5 Then
TempInputRule1.m_iState = 2
Else
TempInputRule1.m_iState = 1
End If
ElseIf SensorVal1 = "Medium" Then
If m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule1.m_iState = 4
ElseIf m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 5 Then
TempInputRule1.m_iState = 3
Else
TempInputRule1.m_iState = 2
End If
ElseIf SensorVal1 = "Low" Then
If m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule1.m_iState = 5
ElseIf m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 5 Then
```

```
TempInputRule1.m_iState = 4
Else
TempInputRule1.m_iState = 3
End If
ElseIf SensorVal1 = "V_Low" Then
If m_pSensor(TempInputRule1.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule1.m_iState = 6
Else
TempInputRule1.m_iState = 5
End If
Else
MsgBox "Debug Log: Error in SensorVal1_Click"
End If
End Sub


Private Sub SensorVal2_Change()
SensorVal2_Click
End Sub

Private Sub SensorVal2_Click()
If m_bInResetMode = True Then
Exit Sub
End If
If SensorVal2 = "" Then
MsgBox "U have not selected from the list "
SensorVal2.Text = "Select"
Exit Sub
End If

TempInputRule2.m_iCondition = 0
m_bCompleteRuleEntered = True

If SensorVal2 = "V_V High" Then
TempInputRule2.m_iState = 1
ElseIf SensorVal2 = "V_V Low" Then
TempInputRule2.m_iState = 7
ElseIf SensorVal2 = "V_High" Then
If m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule2.m_iState = 2
Else
TempInputRule2.m_iState = 1
End If
ElseIf SensorVal2 = "High" Then
If m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule2.m_iState = 3
ElseIf m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 5 Then
TempInputRule2.m_iState = 2
Else
TempInputRule2.m_iState = 1
End If
```

```
ElseIf SensorVal2 = "Medium" Then
If m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule2.m_iState = 4
ElseIf m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 5 Then
TempInputRule2.m_iState = 3
Else
TempInputRule2.m_iState = 2
End If
ElseIf SensorVal2 = "Low" Then
If m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule2.m_iState = 5
ElseIf m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 5 Then
TempInputRule2.m_iState = 4
Else
TempInputRule2.m_iState = 3
End If
ElseIf SensorVal2 = "V_Low" Then
If m_pSensor(TempInputRule2.m_iSensorIndex).m_iNumRanges = 7 Then
TempInputRule2.m_iState = 6
Else
TempInputRule2.m_iState = 5
End If
Else
MsgBox "Debug Log: Error in SensorVal1_Click"
End If
End Sub

Private Sub ComboAndOR_Change()
ComboAndOR_Click
End Sub

Private Sub ComboAndOR_Click()
If m_bInResetMode = True Then
Exit Sub
End If
If ComboAndOR = "" Then
MsgBox "U have not selected from the list "
ComboAndOR.Text = "Select"
Exit Sub
End If

If ComboAndOR = "And" Then
InCombo2.Enabled = True
TempInputRule1.m_iCondition = 2
ElseIf ComboAndOR = "Or" Then
InCombo2.Enabled = True
TempInputRule1.m_iCondition = 1
ElseIf ComboAndOR = "None" Then
ComboAndOR.Text = "Select"
Else
MsgBox "Debug Log: Error in ComboAndOR_Click"
```

```
End If
End Sub

Private Sub OutCombo_DropDown(Index As Integer)
Do While OutCombo(Index).ListCount() > 0
OutCombo(Index).RemoveItem (0)
Loop
iTempNumSensor = 1
Do While iTempNumSensor <= NumSensors
If m_pSensor(iTempNumSensor).m_bInputSensor = False Then
If Index = 1 Then
OutCombo(Index).AddItem (m_pSensor(iTempNumSensor).m_pSensorName)
ElseIf Index = 2 Then
If Not m_pSensor(iTempNumSensor).m_pSensorName = OutCombo(1) Then
OutCombo(Index).AddItem (m_pSensor(iTempNumSensor).m_pSensorName)
End If
ElseIf Index = 3 Then
If Not m_pSensor(iTempNumSensor).m_pSensorName = OutCombo(1) Then
If Not m_pSensor(iTempNumSensor).m_pSensorName = OutCombo(2) Then
OutCombo(Index).AddItem (m_pSensor(iTempNumSensor).m_pSensorName)
End If
End If
Else
MsgBox "Debug Log: Error in OutCombo_DropDown"
End If
End If
iTempNumSensor = iTempNumSensor + 1
Loop
End Sub

Private Sub OutCombo_Change(Index As Integer)
OutCombo_Click (Index)
End Sub


Private Sub OutCombo_Click(Index As Integer)
If m_bInResetMode = True Then
Exit Sub
End If
RuleInitialize
If OutCombo(Index) = "" Then
MsgBox "U have not selected from the list "
OutCombo(Index).Text = "Select"
Exit Sub
End If
m_uNumOutSensor = Index
OutValCombo(Index).Enabled = True
m_bCompleteRuleEntered = False

Do While OutValCombo(Index).ListCount() > 0
OutValCombo(Index).RemoveItem (0)
```

```
Loop

iTempNumSensor = 1
Do While iTempNumSensor <= NumSensors
If OutCombo(Index) = m_pSensor(iTempNumSensor).m_pSensorName Then
TempOutputRule(Index).m_iSensorIndex = iTempNumSensor
If m_pSensor(iTempNumSensor).m_iNumRanges = 3 Then
OutValCombo(Index).AddItem ("High")
OutValCombo(Index).AddItem ("Medium")
OutValCombo(Index).AddItem ("Low")
ElseIf m_pSensor(iTempNumSensor).m_iNumRanges = 5 Then
OutValCombo(Index).AddItem ("V_High")
OutValCombo(Index).AddItem ("High")
OutValCombo(Index).AddItem ("Medium")
OutValCombo(Index).AddItem ("Low")
OutValCombo(Index).AddItem ("V_Low")
Else
OutValCombo(Index).AddItem ("V_V High")
OutValCombo(Index).AddItem ("V_High")
OutValCombo(Index).AddItem ("High")
OutValCombo(Index).AddItem ("Medium")
OutValCombo(Index).AddItem ("Low")
OutValCombo(Index).AddItem ("V_Low")
OutValCombo(Index).AddItem ("V_V Low")
End If
Exit Do
End If
iTempNumSensor = iTempNumSensor + 1
Loop
End Sub


Private Sub OutValCombo_Change(Index As Integer)
OutValCombo_Click (Index)
End Sub

Private Sub OutValCombo_Click(Index As Integer)
If m_bInResetMode = True Then
Exit Sub
End If
If OutValCombo(Index) = "" Then
MsgBox "U have not selected from the list "
OutValCombo(Index).Text = "Select"
Exit Sub
End If

If Index < 3 Then
OutCombo(Index + 1).Enabled = True 'enable the next output sensor combo
End If

Direction(Index).Enabled = True
```

```
If OutValCombo(Index) = "V_V High" Then
TempOutputRule(Index).m_iState = 1
ElseIf OutValCombo(Index) = "V_V Low" Then
TempOutputRule(Index).m_iState = 7
ElseIf OutValCombo(Index) = "V_High" Then
If m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 7 Then
TempOutputRule(Index).m_iState = 2
Else
TempOutputRule(Index).m_iState = 1
End If
ElseIf OutValCombo(Index) = "High" Then
If m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 7 Then
TempOutputRule(Index).m_iState = 3
ElseIf m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 5 Then
TempOutputRule(Index).m_iState = 2
Else
TempOutputRule(Index).m_iState = 1
End If
ElseIf OutValCombo(Index) = "Medium" Then
If m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 7 Then
TempOutputRule(Index).m_iState = 4
ElseIf m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 5 Then
TempOutputRule(Index).m_iState = 3
Else
TempOutputRule(Index).m_iState = 2
End If
ElseIf OutValCombo(Index) = "Low" Then
If m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 7 Then
TempOutputRule(Index).m_iState = 5
ElseIf m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 5 Then
TempOutputRule(Index).m_iState = 4
Else
TempOutputRule(Index).m_iState = 3
End If
ElseIf OutValCombo(Index) = "V_Low" Then
If m_pSensor(TempOutputRule(Index).m_iSensorIndex).m_iNumRanges = 7 Then
TempOutputRule(Index).m_iState = 6
Else
TempOutputRule(Index).m_iState = 5
End If
Else
MsgBox "Debug Log: Error in OutValCombo_Click"
End If
End Sub

Private Sub Direction_Change(Index As Integer)
Direction_Click (Index)
End Sub

Private Sub Direction_Click(Index As Integer)
If m_bInResetMode = True Then
```

```
Exit Sub
End If
If Direction(Index) = "" Then
MsgBox "U have not selected from the list "
Direction(Index).Text = "Direction"
Exit Sub
End If

m_bCompleteRuleEntered = True

If Direction(Index) = "Forward" Then
TempOutputRule(Index).m_iDirection = 0
Else
TempOutputRule(Index).m_iDirection = 1
End If
End Sub
```

Code for Run Program

```
VERSION 5.00
Begin VB.Form RunProgram
  Caption        =   "Form5"
  ClientHeight   =   3195
  ClientLeft     =   60
  ClientTop      =   345
  ClientWidth    =   4680
  LinkTopic      =   "RunProgram"
  ScaleHeight    =   3195
  ScaleWidth     =   4680
  StartUpPosition =   3  'Windows Default
  Begin VB.TextBox OutFile
    Height         =   375
    Left           =   1920
    TabIndex       =   5
    Text           =   "output.cpp"
    Top            =   1440
    Width          =   2295
  End
  Begin VB.TextBox InFile
    Height         =   375
    Left           =   1800
    TabIndex       =   4
    Text           =   "exinput.txt"
    Top            =   480
    Width          =   2175
  End
  Begin VB.CommandButton Cancel
    Caption        =   "Cancel"
    Height         =   495
    Left           =   2880
    TabIndex       =   3
    Top            =   2520
```

```
    Width          =   1095
  End
  Begin VB.CommandButton OK
    Caption        =   "OK"
    Height         =   495
    Left           =   1080
    TabIndex       =   2
    Top            =   2520
    Width          =   1095
  End
  Begin VB.Label Label2
    Caption        =   "Output File:"
    Height         =   375
    Left           =   480
    TabIndex       =   1
    Top            =   1440
    Width          =   1095
  End
  Begin VB.Label Label1
    Caption        =   "Input File: "
    Height         =   255
    Left           =   360
    TabIndex       =   0
    Top            =   360
    Width          =   1095
  End
End
Attribute VB_Name = "RunProgram"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Cancel_Click()
Close
Hide
Form4.Show
End Sub

Private Sub Form_Load()
Rem get the input file name from the sensor/rule input page
InFile.Text = CurDir + "\" + "exinput.txt"
OutFile.Text = CurDir + "\" + "output.txt"
If m_bToBeSaved = True Then
InFile.Text = Form5.InFile.Text
End If
End Sub

Private Sub OK_Click()
Dim PathName As String
PathName = "thesis.exe"
PathName = PathName + " "
```

```
PathName = PathName + InFile.Text
PathName = PathName + " "
PathName = PathName + OutFile.Text
Shell (PathName)
Close
Hide
Form4.Show
End Sub
```

Code for Form 5

```
VERSION 5.00
Begin VB.Form Form5
   Caption        =   "Form5"
   ClientHeight   =   2730
   ClientLeft     =   60
   ClientTop      =   345
   ClientWidth    =   4815
   LinkTopic      =   "Form5"
   ScaleHeight    =   2730
   ScaleWidth     =   4815
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton OK
      Caption        =   "OK"
      Height         =   495
      Left           =   840
      TabIndex       =   3
      Top            =   1920
      Width          =   1095
   End
   Begin VB.CommandButton Cancel
      Caption        =   "Cancel"
      Height         =   495
      Left           =   2640
      TabIndex       =   2
      Top            =   1920
      Width          =   1095
   End
   Begin VB.TextBox InFile
      Height         =   375
      Left           =   2160
      TabIndex       =   1
      Text           =   "exinput.txt"
      Top            =   960
      Width          =   2175
   End
   Begin VB.Label Label1
      Caption        =   "Plese enter the filename: "
      Height         =   255
      Left           =   120
      TabIndex       =   0
      Top            =   960
```

```
    Width          =   2055
  End
End
Attribute VB_Name = "Form5"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Cancel_Click()
Close
m_bToBeSaved = False
Hide
Rem Form3.Show
End Sub

Private Sub Form_Load()
m_bToBeSaved = False
InFile.Text = CurDir + "\" + "exinput.txt"
End Sub

Private Sub OK_Click()
m_bToBeSaved = True
Hide
End Sub
```