
W. R. YU
M.S.
1999

This is to certify that the

thesis entitled

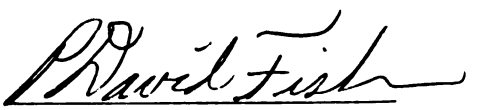
FUZZY LOGIC FOR EMBEDDED SYSTEMS:
DESIGN OF A REFRIGERATOR CONTROLLER

presented by

Weijiang Robert Yu

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical Eng



Major professor

Date 8/11/99

FUZZY LOGIC FOR EMBEDDED SYSTEMS:

DESIGN OF A REFRIGERATOR CONTROLLER

By

Weijiang Robert Yu

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

1999

ABSTRACT

FUZZY LOGIC FOR EMBEDDED SYSTEMS:

DESIGN OF A REFRIGERATOR CONTROLLER

By

Weijiang Robert Yu

Fuzzy logic design has been popular in the engineering domain for decades. Its application ranges from control to decision support technology. Fuzzy logic design overpowers the more conventional design strategies because of the ease with which it can represent linguistic terms and ill-defined problems. Moreover, it is very simple to implement using low cost micro-controller in solving complex and non-linear problems. In this thesis, the motivation and relevance for applying fuzzy logic to embedded systems is examined. The history of fuzzy logic and the concepts of the mathematical foundation that fuzzy logic is built on – fuzzy set theory – is studied. Research has concentrated on the most popular fuzzy logic design methods for embedded systems. Details in constructing fuzzification interface, defuzzification interface, and rule evaluation block are presented. The hardware/software implementation issues are studied. This thesis also provides a demonstration of applying fuzzy logic design to a home appliance followed by results analysis and recommendations.

DEDICATION

This work is dedicated to my parents.

ACKNOWLEDGMENTS

This thesis would not have been possible without the help and support of ECE faculty and staff. Special thanks go to Mr. Carlo Adinata and EE482 Home Appliances Design Team, who provided valuable data for my project, and many thanks to Mr. Peter Leslie Semig, who helped me check the grammar and format of this thesis. My thank also goes to Whirlpool Corporation, who supplied the refrigerator that used in this investigation. In addition, my thanks go to Dr. Diane T. Rover and Dr. Bruce Kim for their critical review of my thesis and their valuable suggestions to help me improve the quality of my thesis. Most importantly, special recognition and thank goes to Dr. P. D. Fisher, whose encouragement, guidance and support were essential for me to finish this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Fuzzy logic has been enjoying remarkable popularity since 1965, when Dr. L. A. Zadeh first proposed this concept [1]. Fuzzy logic theory offers a conceptual framework for the representation of linguistic terms such as "very", "maybe", or "pretty fast". Fuzzy logic design provides a solution to either optimal control problems where the choice of optimality criteria is a matter of experience, or ill-defined control problems where criteria is not quite clear to the Boolean logic based modern computers. Industry experts predicted that fuzzy logic would play an important role in embedded systems and become a multibillion-dollar business [2].

Typical application areas of embedded systems include home appliance, consumer electronics, industry controller, vehicular technology, robotics, power systems and image processing and pattern recognition.

This thesis aims to understand the motivation for fuzzy logic, grasp the comprehensive concepts of fuzzy logic theory, summarize the fuzzy logic design methodology for embedded systems, demonstrate certain fuzzy control algorithms for an embedded system used in a home appliance, and present the results of the design.

1.1 Objectives

The main goal of this thesis is to summarize and show the fuzzy logic design methodology for embedded systems, the most commonly used "fuzzy" algorithms for designing fuzzy controllers and to demonstrate its application in home appliance. Toward this end, some specific objectives were involved and these are briefly described below.

The relevance of fuzzy logic design for embedded systems was to be studied. A refrigerator was to be examined and a refrigerator controller for defrosting system by using fuzzy logic was to be designed. The user interface and energy saving issues of the refrigerator were to be examined and compared with the conventional design.

1.2 Outline

This thesis is organized into five chapters, beginning with this introductory chapter. The chapter is followed by an overview of the history of fuzzy logic, the concepts of fuzzy set, and the important issues on fuzzy logic related concepts such as Frame of Cognition and linguistic hedges. The next chapter describes the relationship between expert system and fuzzy system, examines the structure of a fuzzy controller and the most

popular algorithms used by embedded system design engineers. In the next chapter, chapter 4, two problems that used to be solved by conventional solution in home appliance industry are examined, and two design proposals using fuzzy logic technology to improve the performance of the refrigerator are suggested. Software implementation of the fuzzy defrosting system is demonstrated. The last chapter deals with the analysis of the results of the fuzzy defrosting system. Recommendations on further optimization of the fuzzy defrosting system and software/hardware implementation of the temperature setting feature.

Chapter 2

FUZZY LOGIC

2.1 A Brief History of Fuzzy Logic [3]

The modern logic can trace its origins back to ancient Greek philosophy. Due to the efforts of those famous philosophers, particularly Aristotle, a theory of logic -- the so-called " The Law of Thought" was born [4]. One of the most famous, the "law of Excluded Middle" states that any proposition can be either True or False only. However, Heraclitus who stated that it is possible for things to be both "True" and "Not True" challenged this law.

It was then Plato who indicated that there is a third state (neither True nor False). Then it was Lukasiewicz who first systematically proposed the concept that could be the alternative to the Aristotle's two-value logic [5].

In the early 20th century, Lukasiewicz introduced a three-value logic. He assigned a numeric value to the third state that can be translated as "possible". Later on he found out four-value logic, five-value logic and then he proposed that it is possible to have infinite-value logic.

It is then until recently that the notion of an infinite-value logic was developed. In 1965, Dr. L. A. Zadeh from University of California, at Berkeley published his work about "Fuzzy Sets" which mathematically described the fuzzy set theory [1]. This theory uses the membership function to map the elements of set to the values between 0 and 1 instead of using characteristic function that maps all the elements of the set to the value 0 and 1. He also proposed operations on how to calculate fuzzy logic.

2.2 Fuzzy Sets

Before we get to the concept of the fuzzy set, let us remind ourselves of Boolean Logic, which is a cornerstone of all mathematical tool used. Based on this two-value logic, no matter how complicated an object is, we will assign it to one of two complementary categories, such as good and bad, odd and even. Sometimes it does make sense. For example, we can always classify integer numbers as odd or even. Nevertheless, we are faced with objects that are ill defined and do not have clearly defined boundaries in engineering tasks. Consider the categories such as low pressure, high temperature, small errors etc., which convey a useful meaning that is only obvious

under a certain context. However, the line between the inclusion and exclusion of an object for such categories are not clear. Therefore the conventional two-value logic is not well fit for this kind of problems.

Here is an example of how two-value logic can give rise to undesired phenomena: One piece of wood log does not constitute a pile nor two nor three... On the other hand we will agree that 1 million pieces of log constitute a pile. What is the clear-cut line in between? Can we say that 42,000 pieces of log don't constitute a pile but 42,001 would constitute a pile? As we can see, two-value logic can not perfectly explain this dilemma. The key point of fuzzy sets is that it extends the concept of a set by admitting different grades of belonging, or so called membership. By fuzzy set concept, this wood log problem can be easily solved by allowing for all intermediate situations between complete membership and complete non-membership. The higher the value of the membership of an object M defined by fuzzy set A, the stronger the link of the object to the category described by A. So in the above problem we can define 42,000 pieces of log satisfies the category "pile" at a degree equal to 89%, while 100 pieces of log is characterized by a grade of membership as 2%.

Definition as stated by W. Pedrycz [6]:

A fuzzy set A defined on a universe of discourse X is characterized by membership function

A: X → [0,1]

Where the grade of membership A(x) describes the degree to which x satisfies the class defined by A. A(x) = 1 denotes all the elements are completely belong to A while A(x) =0 denotes all the elements that definitely do not belong to A.

All the properties and laws of crisp set hold except for the exclusion law. However, the exclusive law is an exception:

- $A \cup \bar{A} \neq X$
- $A \cap \bar{A} \neq \phi$

These two relationships simply indicate the under-lap or over-lap between a fuzzy set and its complement. The logic operations defined on a fuzzy set can be described point-wise as:

- **AND(x,y) : min(x,y)**
- **OR(x,y) : max(x,y)**
- **NEG(x) : 1- x**

Where x and y are two grades of membership.

2.3 Linguistic Hedges

Linguistic hedges [7] are special linguistic terms that modify other linguistic term. The terms such as very, high, medium or extreme are examples of linguistic hedges. Linguistic hedges are not applicable to the classic Boolean logic. Terms such as very adult, very horizontal, or very rectangle are meaningless.

Any linguistic hedge, H, can be interpreted as a unary function [7], h, on [0,1]. If h (a) < a, for all a $\in$[0,1], the operation that represents the linguistic term is called **strong**; if h (a) > a, for all a $\in$[0,1], then the operation is called **weak**. The special case that h (a) = a is called **identity**. A strong operation reduces the truth-value of the associate

proposition. A weak operation on the contrary, increases the truth-value of the proposition. An operation H has to satisfy the following self-explanatory conditions summarized in [7]:

- **h(0) = 0 and h(1) = 1;**
- **h is continuous function;**
- **if h is strong, then h^{-1} is weak and vice versa;**

A popular class of functions that satisfy these conditions is

$$H_\alpha(a) = a^\alpha,$$

Where $\alpha \in R^+$, R^+ is a set of positive real numbers and $a \in [0,1]$. When $\alpha < 1$, h is **weak**; when $\alpha > 1$, h is **strong**. So choosing different values of α will let us define and distinguish the meaning of a linguistic term in different context. This is very useful when constructing a membership function for an ill-defined system where nature language that can only be understood in certain context is used. Consider the following example:

P1: Humidity is high,

P2: Humidity is very high,

P3: Humidity is medium high,

we let the linguistic hedges very and medium be represented by the strong operation $h(a) = a^2$ and the $h(a) = a^{1/2}$. So if according to our fuzzy set "High", 50% moisture is considered to be "High" at the grade of 0.8. Then VERY HIGH (50%) = $(0.8)^2 = 0.64$ and MEDIUM HIGH (50%) = $(0.8)^{1/2} = 0.89$, and TRUTH (P2) = 0.64, and TRUTH (P3) = 0.89. We find here that the stronger the assertion is the less true it is, which agrees with our intuition.

2.4 Frame of Cognition [8,9]

When human beings are trying to solve some complicated problems, they first try to classify their knowledge in terms of some general concept and then to find out the essential relationship between these concepts. This kind of Top-Down approach can help us to figure out more detailed solutions from those general knowledge and relationships that connect them. This effort will not help us to find the exact numbers or values we need to achieve our goal, but rather help us to classify some objects into more general categories like sets (fuzzy sets). It is known that human being can only memorize and use very limited numbers of concepts at a time in solving complicated problem, however, the structure of these concepts is important for efficient memorizing, recalling and utilizing.

When linguistic labels are defined by fuzzy sets, they assign a certain property to all the values on the universe discourse. This property is known to be **the grade of belonging**. When the whole universe discourse is defined by labels, some of the regions will then be defined as compatible to the highest extent with the labels. The linguistic label assigned to a certain variable forms a frame of cognition of the variable.

Fuzzy sets {F1, F2,....} form a frame of cognition *A* if:

- *A* covers the universe, i.e. all elements of the universe belong to at least one label with nonzero degree of membership.
- For each individual element, the sum of the degree of belong must be one for all labels that are applicable to this element.

In general, the concept of a frame of cognition is a very helpful concept in the application of fuzzy sets by contributing to an efficient formation of membership functions that are essential to fuzzy logic design. This concept of the fuzzy set is well known for its importance in narrowing the gap between the pure numerical techniques and the pure symbolic methods. The numerical techniques mainly used today by computer stress particularly in precision and lack of the necessary generality. On the other hand, human beings can successfully finish some tasks without even knowing all the exact values of the parameters that involves in their actions. Fuzzy sets come in between like a bridge. Each object in the fuzzy set is described by symbols or so-called **labels** – e.g., small, medium, or large. Meanwhile, these labels are connected with the numerical characteristics of the system through certain semantics. And these numerical characteristics are expressed in the form of membership functions that have numerical grades assigned to them. As we can see later in this thesis, the trade off between the representation generality and the precision can be efficiently satisfied by modifying the parameters of the symbols and the number of the symbol itself. The modification is largely done to the membership function and rule evaluation without any big changes in the design structure.

Chapter 3

CONSTRUCTION OF A FUZZY CONTROLLER

3.1 From Expert Control to Fuzzy Control

Tasks like driving an automobile on a snow-covered roadway, adjusting the water temperature when we are taking a shower, or packing a fragile non-standard-shaped object – while common in our life – continuously create tough challenges to the embedded controlled robot. Let's first take a look at how we accomplish our tasks in our everyday life. When a particular task has to be solved, we first will – with our goal in

mind – examine the current situation we are in and try to get as much information as we can. Based on the information we have, we try to recall the entire set similar situations we have been in and their solutions from our experience (our memory serves as a huge knowledge base in our brain). Then with the help of the feedback from the result of our control actions and our goals we quickly achieve the best or close enough result to our goal within a few actions. Consider the example of controlling the temperature in a building. People in each individual room tend to have different temperature preferences. Hence, human being operators have to be replaced in these processes of control by computers or embedded micro-controllers. These controllers must mimic the decision-making skills of the room occupants.

A comprehensive module has to be built to undertake all the control activities. However, just as human beings can not describe an object if they have no prior knowledge of it, the controller cannot be constructed to replace human beings when it does not incorporate a complete control knowledge base. The traditional modules used in engineering all have the following premises [6]:

- The system has to be known; if there are some uncertainties with the system, the control result gotten by the human-replaced controller can not be optimal.
- The known system has to be clearly and completed defined by mathematical models.
- The goal of the control has to be specified in some direct system related variables in the term of clearly defined formulas. These variables are also called performance indices.

Unfortunately, when the complexity of the system increases, incorporating these assumptions completely into the control model is not realistic. This is due to the fact that

some of the systems are nonlinear or some factors of the systems are non-stationary, such as time variant systems. A more common problem today is not only the difficulty at finding such a complete model, but also the dilemma that a more precise model usually leads to more complicated multi-valued functions with more undetermined parameters to be estimated. Even after we overcome all these troubles and finally find such models, we can only keep them as nice theoretical solutions in the textbook. Given the current technology, it is almost impossible to use embedded systems with limited on-board resources to implement the complicated equations that need extensive calculation time while still satisfying the requirement of real-time control.

Moreover, in a real world task, it is likely that we can't specify the performance index that is supposed to be minimized in a numerical manner; i.e., the degree to which different people feel comfortable about the room temperature is different. Sometimes, a complete mathematical model can not even be used to achieve the desired performance index. This usually happens when more than one goal needs to be accomplished within the control actions and where these goals are contradictory. A good example for this is the air conditioner. Some people want the room to be kept cool while they also want the air conditioner to be energy efficient.

This will explain the situation in the human operator controlled system (or so called expert system) where operators can complete the task very well but can not explain too much on why and how they achieved the optimality in the way they handled the control activities. The answer in general is the effective use of the symbolic computation, which is good at interacting with ill-defined problems or linguistic hedges, as well as numerical values. The expert system owes its success to engineering heuristics.

Heuristics plays an important role in achieving the compromise of partial criteria of contradictory requirements. It is well accepted that the fuzzy controller is one of many kinds of expert systems, which has a computerized inference mechanism to represent vague human decision.

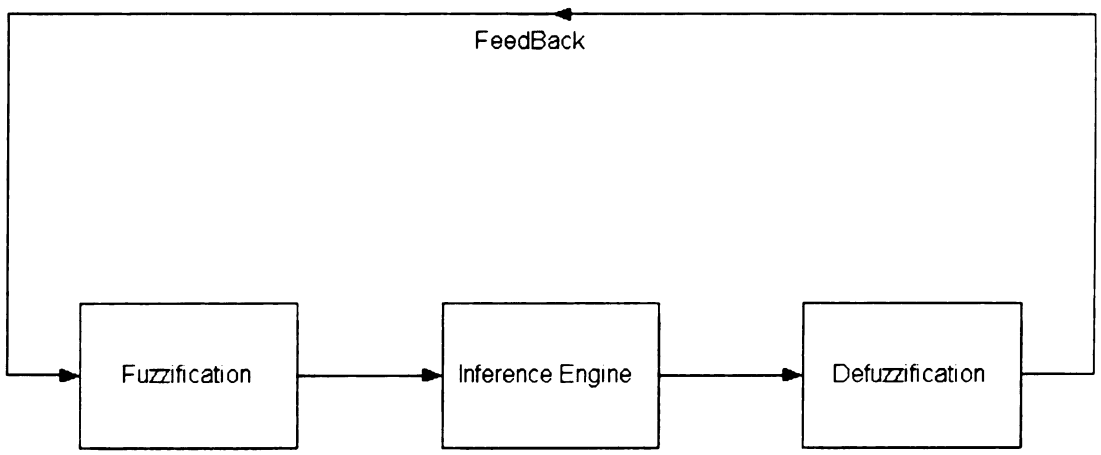


Figure 3.1: Structure of a Fuzzy Controller

3.2 Structure of A Fuzzy Controller

It is worth emphasizing that the purpose of fuzzy control is to analyze and process fuzzy information using a non-fuzzy reasoning scheme. Its fuzziness comes from the nature of the ill-defined problem (e.g., problems that are not clearly defined, such as what the control action of the air conditioner should be if the room temperature is high), the vague representation of the goal (e.g., what constitute energy efficiency), and the way the knowledge base is built on. All this natures determine the structure of a fuzzy controller. A typical fuzzy controller has three parts, as shown in Figure 3.1:

- Fuzzification
- Rule-based knowledge base (Rule Evaluation or so called Inference Engine)
- Defuzzification

3.2.1 Fuzzification

Fuzzification process – so called **input interface** – involves domain transformation. Some technical terms, as shown in Figure 3.2, in this process are:

Labels: names that are used to describe the membership function i.e. short fast.

Universe discourse: the range of all the possible value that is applicable to the system.

Crisp inputs: distinct real time input values.

Fuzzy inputs: the degree, on a 0 to 1 scale, to which the crisp input agrees the membership function.

Membership function: each membership function is a fuzzy set. Each membership function is a function that maps eligible crisp input value to a fuzzy input that satisfy the concept of the term of the labels that describe the membership function.

Scope of each label: The range of crisp inputs that the membership function will map.

With all of these concepts, the membership function is the key for the fuzzification process. As we will see later in this chapter, the way rule-based knowledge base is assembled determines that at least one label must be assigned to each membership function. These labels are always linguistic variables or linguistic representations of technical figures often used by human beings. For example, the technical figure "Temperature" can have a linguistic interpretation {hot, medium warm, normal, medium cold, freezing}. From psychology research on human beings, it has been shown that

humans use their short-term memory to process the technical figures and the short-term memory can handle only up to seven terms at a time. As it is general accepted as a rule, real world engineering always use symmetric labels that have their number as three, five, or seven.

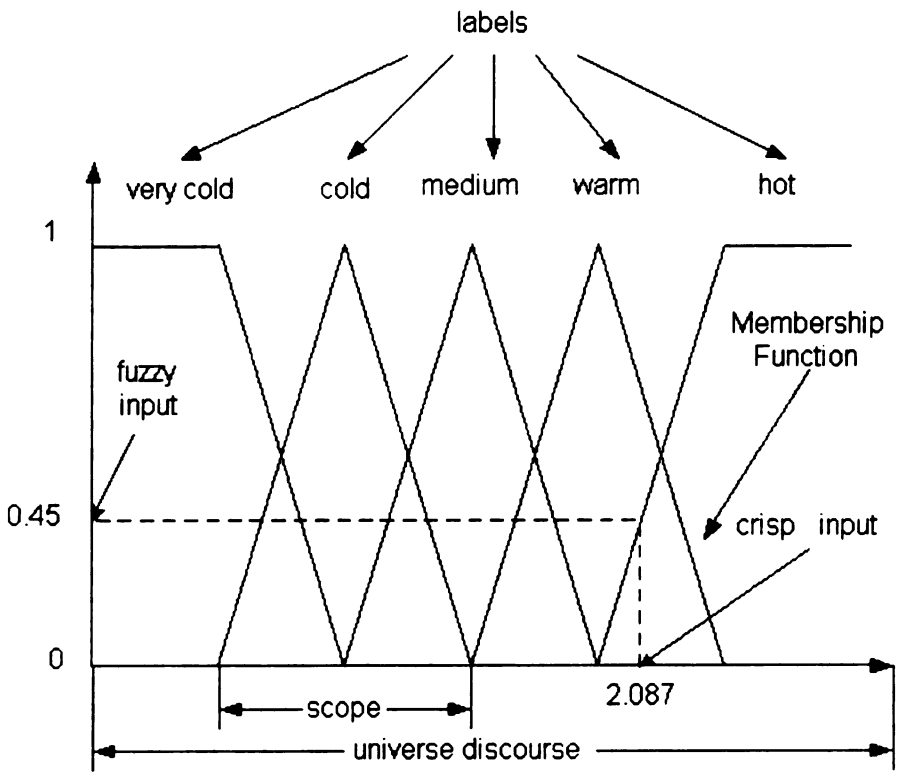


Figure 3.2: Illustration of Fuzzy Logic Terminology

In all these concepts, the membership function is the most important in the fuzzification process. Membership function is established to assign numerical values to the linguistic variable. It is this membership function that makes the fuzzy controller a powerful but, also, a flexible tool to cope with the linguistic variable in the process of control while still precise enough for computer implementation. The membership function is unlike the characteristic function used in Boolean logic, where labels fully

agrees the characteristic function on one side of the clear cut boundary and doesn't agree at all on the other side. Instead, the crisp inputs within the scope of the membership function changes gradually from fully inapplicable to fully applicable.

Many different shapes of membership function have been proposed by fuzzy logic research. Only four are used at practice (see Figure 3.3): Z-type, Λ-type (lambda), Π-type (PI), and S-type. These types are all normalized and therefore, their maximum is always 1 and minimum is always 0. Other kind of membership functions are discarded by engineers at practice either because they are too complicated to implement for embedded system or because they can be replaced by Z-type or S-type functions. Sometimes people use cubic spline shape function to get better performance. This spline membership function is still based on the previous four types but connect the maximum and minimum using spline lines instead of straight lines. The improved performance, however, is achieved by increasing the complexity of the implementation of the membership function.

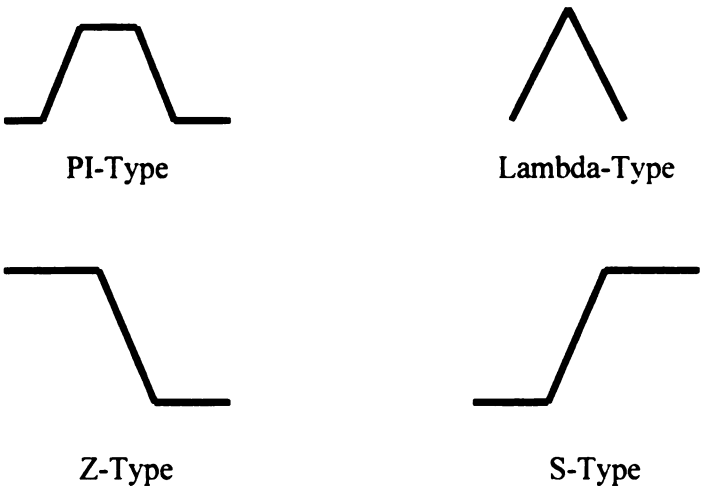


Figure 3.3: Different Shapes of Membership Functions

During the process of fuzzification, the membership function will take a real time input value and assign a unique grade value for each given label. After the fuzzification, there will be a fuzzy input for each label corresponding to the current crisp input. It is important to know that the sum of the fuzzy inputs for all labels is 1.

Usually the computation for the membership is the most time consuming part. However, by using the standard membership function, an optimal fuzzification algorithm exists [10]. In this algorithm, membership function is represented by two base points and two slopes. As indicated in Figure 3.4 that is shown below, since A and B are always equal to 0 and 1, respectively, the generated code will only store this two points. In addition, the code stores the slopes of the two straight lines that across A and B.

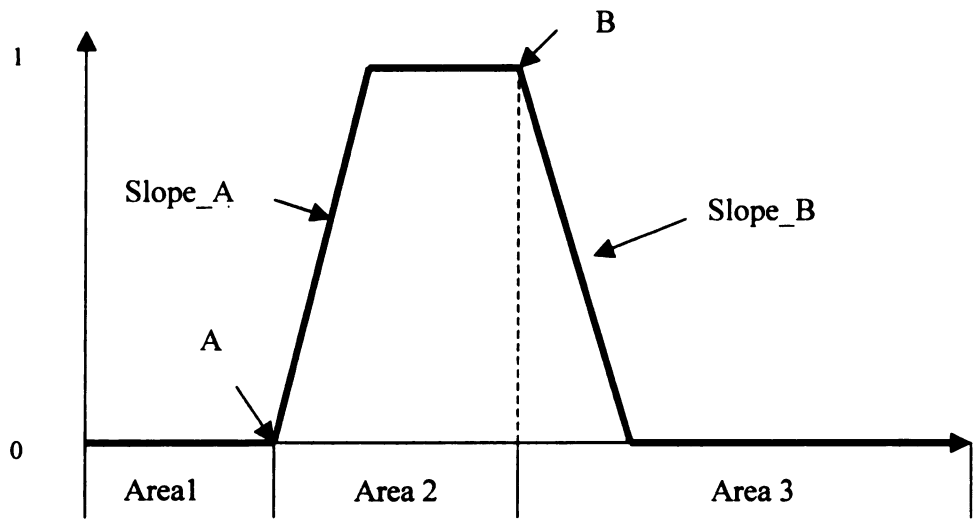


Figure 3.4: Values Stored for Optimal Fuzzification

Therefore, the membership in the figure for three different areas are:

Area 1: zero.

Area 2: min {(crisp input – A)* slope_A, 1}

Area 3: max {1 – (crisp input – B)*slope_B, 0}

Since only comparison, subtraction and one-step multiplication is used, the speed of this optimal algorithm is five times faster than the four points representation of the PI-type membership on a normal micro-controller with no multiplier/divider.

3.2.2 Rule-based knowledge base (Rule Evaluation)

After we get the fuzzy input we need, we will proceed to the second part of the "fuzzy" processing – **Rule Evaluation**. The knowledge base is made up of if-then statements with linguistics variables. For example, a typical rule will look like "IF the Temperature is Hot and the Humidity is Wet THEN the cooling time is Long". The concept of rule evaluation is to determine what the control action should be based on the linguistics rules made by experts given the fuzzy input value. The rules are written in nature language, however, it is confined to a set of predefined linguistics terms and observes a strict syntax. The syntax is in the form of:

IF antecedent_1 And antecedent_2 And ...

THEN consequent_1 And concequent_2 ...

Where each antecedent is in the form of:

Crisp Input Variable is Label

For example, Temperature is Hot. Here Temperature is the name of the crisp input and Label is Hot, which is one of a few labels that describe the input membership function of Temperature.

The consequent takes a similar form:

Crisp Output Variable is Label

For example, Cooling Time is Long. Here Cooling Time is the name of the crisp output and Long is one of many Labels that describe the output membership function of Cooling Time.

The term "And" is fuzzy operator that will be applied to get the minimum value from all the antecedents. Actually, fuzzy rules presented originally allow other fuzzy operators such as fuzzy OR and NOT as we mentioned in previous chapter. However, when rules are implemented, normalized rule representation is used. For rules like "IF A OR B THEN C" will be rewritten as "IF A THEN C" and "IF B THEN C". For rule like "IF NOT A THEN B" we still take this rule as it is in the form of "IF C THEN B". But we know if the truth value of A is 0.7, then the truth value of C which is NOT A is 0.3. Some of rules doesn't depend on any input variables, such as "Cooling Time is Long". This type of rule is called assertion. This kind of rules is not very common in the control tasks, but sometimes it happens. It has the same properties when we calculate the truth-value of rules.

Knowledge base is a rule block, which includes all the assertions and normalized rules. One advantage of fuzzy logic knowledge base is that it is all made up of linguistics terms like natural language. So the expert doesn't have to know anything about computers to write the rules. And when the programmer tries to implement the knowledge base, he doesn't have to have any prior knowledge of the expert system to implement the rule base in the way the expert wants as long as rules are written in the required format.

During the rule evaluation, fuzzy inputs of each label, which are gotten from fuzzification will be plugged into corresponding rules that have the same crisp input

variable and labels. Then the truth-value of each individual rule will be calculated using the Min-Max technique. The truth-value of each rule will be used to determine the truth-value of the "THEN" part consequent. The truth-value of each consequent is called fuzzy output. In the Min-Max technique, the minimum truth-value of all the antecedents of each rule will be written down as the truth-value of the current rule. Then after all the rules have their truth-values, we find the truth-values of all consequent in every rule and for each consequent the fuzzy output is the maximum value of the truth-values of this consequent in all rules. For instance, if we got the fuzzy inputs of each labels {Hot, Medium, Cold} describing the crisp input variable Temperature are {0.8, 0.2, 0} and the fuzzy inputs of labels {Wet, Medium, Dry} that describe crisp input variable Humidity are {0.6, 0.4, 0.}. And the rule block has four rules (for the sake of simplicity, we also denote Truth (#) as the truth value of rule #):

1. IF Temperature is Hot (0.8) and Humidity is Wet (0.6)

THEN Cooling Time is Long. **Truth (1) = min (0.8,0.6) = 0.6**

2. IF Temperature is Medium (0.2) and Humidity is Dry (0.0)

THEN Cooling Time is Short. **Truth (2) = min (0.2,0.0) = 0.0**

3. IF Temperature is Cold (0.0) and Humidity is Medium (0.4)

THEN Cooling Time is Short. **Truth (3) = min (0.0,0.4) = 0.0**

4. IF Temperature is Medium (0.2) and Humidity is Medium (0.4)

THEN Cooling Time is Long. **Truth (4) = min (0.2,0.4) = 0.2**

So the fuzzy output for label "Long" that describes crisp output "Cooling Time" is max (0.2,0.6) = 0.6, and the fuzzy output for label "Short" that describes crisp output " Cooling Time" is max (0.0,0.0) = 0.0. So given the membership functions assigning to each crisp variables and the current rules, the rules inference generates the fuzzy outputs {0.6,0.4,0.0} for labels {Long, Medium, Short} describing crisp output "Cooling Time". It is pretty clear that a control action will be taken for a time length somewhere between "Long" and "Medium" as we defined. But we only have a general idea of what the time length should be. In order to get the exact value that computer or micro-controller can understand we need to defuzzify the fuzzy outputs and find the crisp value of the output variable. In real world task, rules can have different weight. So some rules may have higher priority than others, some rules may never fire.

3.2.3 Defuzzification

The conversion from linguistics fuzzy output to a real non-fuzzy value is called defuzzification. The output membership function uses exact the same types of membership functions as input membership function does. In practice, singleton membership functions are also used for output membership function for simplicity. There are different defuzzification methods given the meaning of the linguistics terms. In general, there are two big categories as being called by C. V. Altrock: **Best Compromise** and **Most Plausible Result** [10].

- **Best Compromise:**

In the previous example, fuzzy output is {0.6,0.4,0.0}. Let's assume that the typical values of each labels for output Cooling Time are:

Long 30 minutes

Medium 15 minutes

Short 0 min

Then for the best compromise result, as indicated in Figure 3.5, we calculate the expectation of Cooling Time:

$$30 \times 0.6 + 15 \times 0.4 = 24 \text{ minutes}$$

Defuzzification methods that are used for best compromise include:

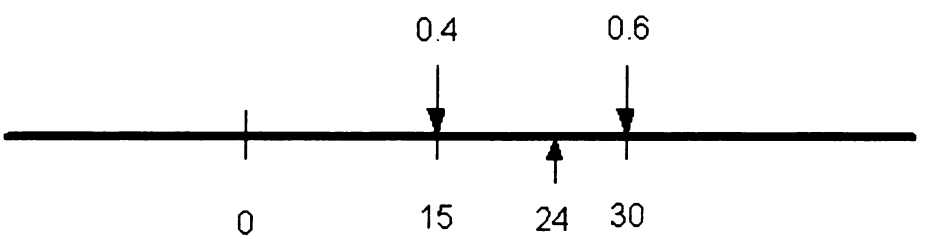


Figure 3.5: Illustration of Best Compromise

Center Of Area (CoA) / Center Of Gravity (CoG):

This method first truncates the values of each label, which are above the fuzzy output of that label and then try to find the balance point of the membership functions or so called center of area / center of gravity. This balance point indicates the best compromise (see Figure 3.6).

CoA/CoG works fine with those whose membership functions have identical shape. If two membership functions have different shapes (the areas of membership function are

different), then the one that has bigger area will have a greater impact on the output than the one that has smaller area when both have the same degree of validity. But, as we know, having larger area simply means there are more values fit in the category described by the label than the one with smaller area.

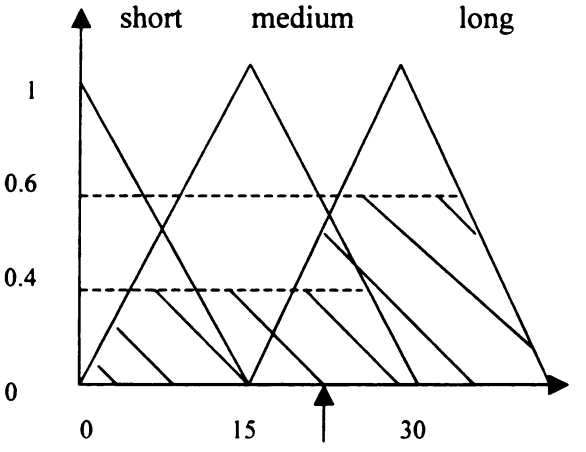


Figure 3.6: CoA Defuzzification

Another disadvantage of CoA/CoG method is its large amount of computation involved during the process. This is mainly because the effort spent on calculating integration of the truncated membership functions. For this reason, some fuzzy logic processors use what so called fast CoA. We can have a pretty good estimation of the result by taking a few sample points over the membership function and superimpose these samples. Since the overlapped areas of membership functions are neglected during the calculation, this approach can only be seen as an approximation of CoA.

Center of Maximum (CoM):

This is an even simpler version of CoA. The example given at the beginning of this section uses this method. This approach only takes the typical value of each membership function rather than calculates the area under each membership function. Notice that if the output membership function is singleton, then CoA/CoG is the same as CoM. CoM is a very popular solution in real world engineering due to its high computation efficiency.

- **Most plausible result**

Sometimes best compromise doesn't apply to some problems. For instance, a robot is controlled by an embedded micro-controller. The output labels are {steer left 30°, stay straight, steer right 30°}, and the fuzzy output are {0.5, 0, 0.5} indicating there is an obstacle right in front of the robot. Under this circumstance, the robot should turn to either left or right, both will be fine. However, if we are using the best compromise method, we will get the steering angle 0, as shown in Figure 3.7. This is obviously wrong.

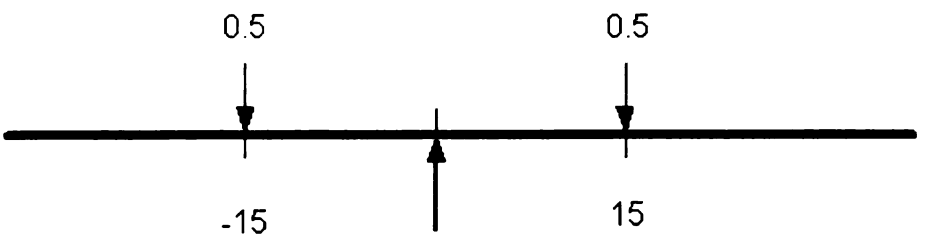


Figure 3.7: Illustration of Most Plausible

The approach solving this situation is called Mean of Maximum. Instead of finding the balance point, MoM finds the most valid term and takes the typical value of that term as the result.

A very important property of defuzzification is continuity. If small change of input doesn't cause a sudden change on any variables at the output end, then we will say the defuzzification method used is continuous. In a closed-loop control process, the last thing we want to happen is the output jump abruptly responding to small changes of inputs. This will cause the output oscillation and therefore system instability. The CoA/CoG and CoM are continuous methods because the result gotten from these methods are the compromise of all the possibilities, therefore abrupt jump can never happen. The MoM, however, can sometimes come up solutions that are completely opposite. So in the real world, an integrator will be connected to the fuzzy controller to keep the output continuous. The "best compromise" methods are used mostly in quantitative problems such as control, whereas the most plausible methods are used mainly in decision support. A comparison between all the methods mentioned above is listed in Table 3.1 below.

	COA, COG	FAST COA	COM	MOM
Categories	Best Compromise	Best Compromise	Best Compromise	Most Plausible Result
Membership Function shape	No good for MF with different shape	No good for MF with different shape	Good	Good
Computation efficiency	Very low	OK	High	Very High
Continuity	Yes	Yes	Yes	No
Applications	Quantitative	Quantitative	Quantitative	Decision making

Table 3.1: Comparison of Defuzzification Algorithms [10]

3.1 Issues of Stability and Verification

The issue of stability has been argued since the fuzzy logic concepts are proposed [11]. Whether or not fuzzy logic is a viable way to enhance the performance achieved by conventional approach has been challenged by lots of critics. The focus is on whether there is a mathematics proof of the stability. It is known that the stability theory of the conventional control completely suffices the fuzzy systems [10]. Fuzzy systems that have the properties of deterministic, time invariant and nonlinear are classified in control theory as nonlinear multi-band controller. Therefore, all the stability analysis that applicable to the conventional nonlinear multi-variant controllers are applicable to the fuzzy controller. But since most fuzzy systems have very complex nonlinear characters, it is almost impossible to find a stability proof. Normally, a conventional approach that has the same complexity is also impossible to get a stability proof.

Chapter 4

DEMONSTRATION AND ANALYSIS

4.1 Introduction

In this chapter, an application of using fuzzy logic to improve the performance and the energy consumption of a home refrigerator will be demonstrated. Problems will be stated and a solution using fuzzy logic algorithm will be suggested and designed. The use of fuzzyTech [12] software will be demonstrated and analysis and recommendations will be given.

Home appliances have been a very important part of people's life today. People are dealing with them in their everyday life. Companies have tried their best to bring the most easily to use but least energy-consumed appliance to the market to attract the customers. Thanks to today's technology, they can offer a lot more dreams features to the

customers than they could five years ago [13]. However, home appliances are not industry machines. Their costs have to be as attractive as their performance and they have to have a high-level friendly user interface to satisfy the "non-technical" users. Fuzzy logic is an innovation technology that allows implementing complex and intelligence functions into embedded system. It has the advantages of implement non-linear and adaptive control using limited system resource such as an 8-bit micro-controller. In this demonstration, we will first examine a typical home appliance --- refrigerator from the point of view of energy saving and user interface.

4.2 A Brief Review of the Structure of Refrigerator [14]

A refrigeration system, as shown in Figure 4.1, consists of a compressor, condenser, dryer, capillary tube, evaporator and refrigerant. The compressor pumps evaporated Freon gas from the evaporator coil and then compress it. When the Freon returns to a liquid state, it gains heat. The heat is then released outside the cabinet. After passing the condenser, the refrigerant enters the capillary tube. This is a small metering device that only allows an exact amount of liquid refrigerant to flow through before entering the large tubing of the evaporator. As the refrigerant leaves the small capillary tube, the sudden increase in tube size of the evaporator causes a drop in pressure that allows the fluid to evaporate and cool down. Now it gains heat and ready to be cycled again. See Figure 4.2 for the heat transfer illustration of refrigerator.

As the cycle continues, everything inside the food compartment would eventually freeze. To control the temperature, a thermostat is used to monitor temperature at the evaporator and control the power to the compressor. As the temperature inside the

refrigerator rises, the thermostat turns on the compressor. The thermostat can be adjusted to control the desired refrigerator temperature by a knob as the one we see in most of the refrigerator.

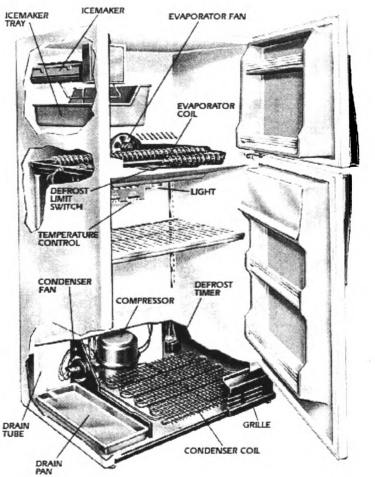


Figure 4.1: Basic Structure of a Refrigerator [14]

During this process, the refrigerator not only picks up heat from food, but also gets moisture from warm air caused by door opening. The moisture then forms the frost on the evaporator. As frost accumulates to a certain point, it reduces the evaporator's

effectiveness. In order to keep the system operating properly, the refrigerator needs periodic defrosting.

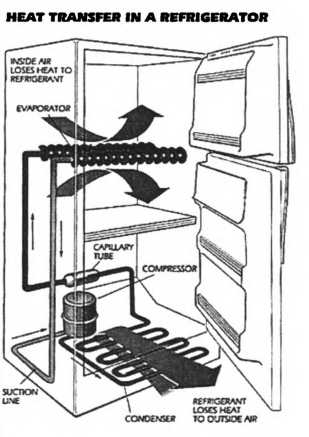


Figure 4.2: Heat Transfer Illustration of Refrigerator [14]

There are three types of defrosting systems in use today--manual, cycle and automatic (no frost). In the manual system, freezer defrosts by turning off the power and let the ice melt. In the cycle system, an electric heater is activated when the compressor is off to melt ice if there is any (freezer section still has to be done manually if there is a heavy buildup of ice). An automatic defrost system consists of a low watt heater, a thermostat to turn on and off the heater, and a timer to control the defrost cycle. As frost

accumulates, the timer starts the defrost cycle. Then, the compressor will be turned off and the heater will be turned on. After the frost cycle is finished, the thermostat turns off the heater and the timer allows the water to drain to a defrosting pan before starting the compressor.

4.3 Problems with Refrigerator

In today's home appliance industry, one most important issue is energy saving [15,16]. The refrigerator is the most common home appliance and is one of the biggest users of electricity in most homes. Depending on its age and size, the refrigerator can account for as much as one third of a non-electrically heated home's electricity usage. They account for nearly 20 percent of residential electricity [15]. The energy saving goal is not to cut the power consumption to zero, but to eliminate all the possibilities of energy wasting. A tremendous extra power will be needed when the frost starts to accumulate in the freezer. The frost accumulation reduces the evaporator's effectiveness, and the refrigerator gets warm. As mentioned above, we have three types of defrost system. The first two types of system are no way user-friendly and therefore are out of consideration. The automatic system is the most popular one in the market because of its user-friendly interface and the no-frost result. But after taking a closer look at its defrost control mechanism we found most of the unnecessary energy waste are coming from the defrost system.

In spring 1998, a characterization test of a home refrigerator was conducted by one of the Design Teams of the computer system design course in the Department of

Electrical and Computer Engineering, Michigan State University. The result [17] has shown that there will be a huge temperature jump whenever the defrosting cycle kicks in. when the defrosting cycle starts, the compressor was turned off and the heater was turned on. The refrigerator with automatic defrost system defrost twice a day with each cycle lasts around one hour. Defrost cycle starts kicking in everyday at a fixed schedule. This is mainly due to the fact that the designer of the defrosting system can not predict the time of the accumulation of frost. So no matter if there is frost, the compressor will be turned off and the heater will be on when defrost cycle starts to guarantee a no-frost. During the defrost cycle, the temperature in the freeze will go up to 15°C from -18°C for about one hour. This unwanted temperature rise is no good to the food. And since when the defrost cycle is off, the compressor has to use extra power to cool the temperature down to its normal, energy waste will occur. In another word, if the customer just went grocery shopping not long before the defrosting cycle starts, then the refrigerator will get warmed up for an hour when it most needs at the moment is cooing down. Moreover, in the case that the door of the refrigerator has been keeping closed for a long time such as when the customer is away on vacation, refrigerator will still defrost twice a day even there is little frost.

Another common energy waste comes from the customers themselves. For most refrigerators, there is a knob for temperature control. However, a lot of customers don't know what the right temperature range the refrigerator should stay in to keep the food fresh and to have less energy wastes. Sometimes, when there is a big load increase, i.e. after grocery shopping, people tend to turn down the temperature setting to get a large cooling effect. Because of this and lacking of the knowledge on what is the right

temperature, people tend to turn the temperature lower than necessary, and most of the time they forget to turn it back up when the right temperature is reached and the food is over cooled. Before the temperature setting is corrected, the extra energy used for the unnecessary cooling is wasted.

4.4 Solutions to the Problems

The solution for the defrosting system to avoid unnecessary energy consumption is adaptive defrost control [16], which removes the frost only when it is necessary. We have noticed that two main factors that can cause the cumulating of frost are humidity and temperature. Temperature changes in the freezer is largely caused by the frequency of door openings and the length that the door is opened (It is shown by surveys that the normal door open time is 30 to 90 seconds). So we can say the defrost cycle is a function of the humidity and the door open time. And it is not linear. In a traditional non-linear implementation, either a complete but complicated mathematical model has to be generated or a look up table has to be provided. As the precision requirement increases, both methods become unrealistic for a real time control using low cost micro-controllers. So a two inputs fuzzy controller will be designed to overcome these difficulties. The fully documented fuzzy defrosting controller design is provided in Appendix A.

As for solving the temperature setting problem, we first look at the mechanism of the compressor. Since the control of compressor is hysteresis control, which set the upper and lower limits for the temperature. The compressor is on only when the upper limit is exceeded and is turned off when the lower limit is hit. Big hysteresis setting delayed the

compressor's switching activities while narrow hysteresis setting increases the frequency of the switching activities. When the compressor switches too frequent, a lot of energy will be used. However, when there is a time we need to start the cooling right away or we have some food that have to stay in a strict temperature range to stay fresh, we have to change the hysteresis setting lower. As mentioned in the previous chapter, fuzzy logic has its advantage of reconciling conflicting requirements. The design idea of this fuzzy controller to improve the temperature setting performance is presented below. Its hardware and software implementation need further work and will be recommended in Chapter 6.

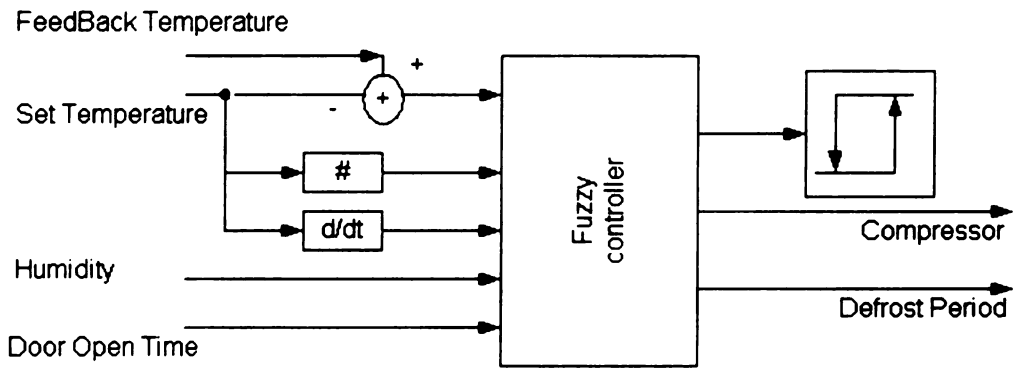


Figure 4.3: Block Diagram of Fuzzy Controller for a Refrigerator

The Temperature Setting fuzzy controller is to have three inputs: difference between the set temperature and the temperature in the refrigerator (or so-called feedback error), changes of the set temperature, and the set temperature that is differentiated by time. The output of the fuzzy controller goes to control the hysteresis and the compressor. When the feedback error is big, fuzzy controller will send signal to the compressor so that the desired temperature will be reached faster, and at the same time set the hysteresis big, so disturbances will not cause unnecessary on-off switches during this time. The time

differentiated set temperature is used by the fuzzy controller to remember for the user to turn the temperature back up when the desired temperature is reached. This input will be zero after a given time if the user doesn't modify the set temperature any more. The number of changes of the set temperature is used to understand if the user want to set the temperature specifically in a certain range. If the user wants to set the temperature precise, he tends to adjust the setting more than once. When this happens, the hysteresis will be set relatively small so the activities of the compressor will be more sensitive to the temperature changes. The completed design block diagram of the refrigerator fuzzy controller is given in Figure 4.3.

4.5 Software and Hardware Development Tool

4.5.1 Hardware platform

The target platform for developing fuzzy logic ranges from normal low cost 8 bit micro-controllers with limited on board RAM and ROM to 64-bit RISC processors. The development of fuzzy logic processor has experienced three phases:

- Analog Fuzzy Processor

The analog fuzzy processor is made up by fuzzy logic gate. A voltage signal ranging from +5V to –5V serves as the membership degree from 0 to 1. This type of processor has never made its way out of the lab, mostly due to its inflexibility in implementation. Any changes of the rules or membership functions require the re-design of the hardware.

- Digital Fuzzy Processor

A digital fuzzy processor is a conventional digital processor a fuzzy instruction set. This type of process can only support fuzzy computation. Two processors were used. One host micro-controller that handles all the peripherals and I/O, and pre-computation, while the fuzzy processor working on the fuzzication, rule evaluation and defuzzification. Both processors are connected either by a series communication port or a dual ported RAM.

The host micro-controller will get the crisp input and send to the fuzzy processor. After the fuzzy processor finish computing, it will send the result back. Because of the overhead of communication, the speed advantage of fuzzy processor is compromised. And since this digital fuzzy processor usually uses up to three chips (if includes and dual ported RAM), the cost is very expensive. In addition, programming for the communications needs extra efforts and extra interrupts in order to synchronize both processors.

- Integrated fuzzy logic processor

In this phase, manufacturers made it possible to integrate both processors within one Chip. This effort reduces the communication overhead remarkably. There are two approaches: An earlier approach treated the fuzzy logic processor just as another cell or memory. This approach requires an additional silicon area and a full functionality of ALU (Arithmetic Logic Unit). The latest approach is to enhance the instruction set of the conventional micro-controller. It keeps the micro-controller's own ALU and adds some additional functions to it. This approach reduces a great deal of areas that the earlier approach requires and the code of the micro-controller is much more compact since all the existing instructions and registers can be used by the fuzzy processing too. This fuzzy

processor can be also used as a general-purpose micro-controller. Some manufacturers even add hardware supports to the fuzzification, rule evaluation and defuzzification to speed up the computation even more. Since no re-design is needed for the micro-controller, this integrated fuzzy processor has great potentiality. Manufacturer such as Motorola has already put its new series micro-controller (i.e. MC68HC12) with fuzzy logic module to the market [18, 19, 20].

4.5.2 Software development

Good design software should offer good combinations of design, simulation, optimization, testing, verification and implementation. There are mainly three types of software development methods. The first one is just using any program language and codes the whole fuzzy system. The second one is to use a pre-compiler to translate the text version of the Fuzzy Logic Description Language or FTL (Fuzzy Technology Language) into a programming language, i.e. ANSI C. The first one usually need re-program a large portion when changes of the fuzzy system are made while the second one is hard to debug due to the text representation. The third and the most popular one is called virtual design interface. The whole design is a matter of drag-and-point graphic implementation. The FTL or any programming code can be generated for specific hardware platform by the code generator using this kind of virtual design software too.

The graphic design software tool I will be introduced is fuzzyTech. The author will demonstrate the fuzzy defrost system using this software.

4.6 FuzzyTech Software [12][21]

- Linguistic Variables Editor

The editor is used to define the linguistic variables and their membership functions. Membership functions are always defined point-wise using definition points that can be easily dragged and positioned by the mouse. There are two alternatives exist to connect the definition points: L-shape and S-shape. For S-shape, the user can specify the asymmetry factor in the range from 0 to 1. For standard membership functions, only the positions of the maximum points are required. The software will automatically generate all the definition points.

- Interface Options

For input interfaces, three algorithms are supported. The first one, Fuzzy Input. Fuzzy input indicates that all the values are inputted into fuzzy control system as a vector of the grade of membership belongings. The second one is compute membership function. This is the most popular fuzzification method used in almost all the applications. As mentioned earlier, the method only stores the definition points and slops and computes the fuzzification at run time. For a few micro-controller, the "look up membership function" is also supported. The computation of the fuzzification is completed in the compile time and stores in a table in the code. This method causes big code size, but is faster overall if the computation for the fuzzification is too complex to do during run time.

For output interface, different defuzzification methods exist too. "Fuzzy Output" outputs a vector of the membership degree of the results of inference. Then there are CoM, CoA, and MoM. For more details, refer to chapter 3. Among those, CoM is the most often used algorithm. There is also one method called BSUM, which represents Bounded Sum. This method uses the bounded sum rather than the maximum operator to aggregate the individual areas. So no overlap area will be computed more than once.

- Rule Editor

Two alternatives of generating rules exist. One is called the spreadsheet rule editor. IF-THEN rule statements can be generated automatically. Users can configure the rule block the way they want it to be created.

1. Alpha cut: delete all rules with the weight less than the alpha value inputted by user.
2. Set DoS: sets the weights of rules to a fixed value
3. Create full rule block: creates rules for each combination of terms. For each possible combination of input variables, a rule for each output variable is generated.
4. Created partial rule block: creates rules for each combination of terms. However, it is different from the "created full rule block" in that the user has to specify the output. The rule that is not specified with output will be considered incomplete and will be erased when the editor is closed.

The other rule editor is called matrix rule editor. This editor is preferred when designing a complex system. The variable terms will be put into a matrix with the degree of membership belongings indicated by the shade of gray. Non-firing rules will be identical to the zero-weighted rules.

Chapter 5

ANALYSIS AND RECOMMENDATIONS

5.1 Introduction

In this chapter, the use of interactive debug of fuzzyTech to analyze the defrosting fuzzy control design will be introduced. The result will be demonstrated and analyzed. Further hardware implementation and software tuning will be recommended.

5.2 Interactive Analysis

In fuzzyTech, after the initial design of a fuzzy controller is done, we can debug and test the design using fuzzyTech's interactive debug tool. Interactive debug mode can be turned on by going to the menu Debug-> Interactive. In the debug mode, any input can be selected and its value can be changed within its universe discourse. This debugger can

be used as a real time data feeder. Designers can then observe the output value and evaluate the result.

There are also three analyzers that can be turned on when the system is in debug mode.

- Transfer Plot

The transfer plot (Figure 5.1) will let users analyze input /output characteristics. When the analyzer is open, two inputs and one output can be selected and resolution can be set over there too. Two input variables are plotted in the horizontal and vertical axes while the value of the selected output variable is indicated as plot color. The color bar on the lower part of the window shows the color-value relation.

- 3D Plot

3D Plot (Figure 5.3) shows the input/output characteristics of the fuzzy system. It allows a plane that is drawn in the control space. Two inputs will be drawn in the horizontal and vertical axis while the value of the output will be indicated as the plot color in the control space.

- Time Plot

Time Plot (Figure 5.2) lets the user to see the time response characteristics of the fuzzy system. User can add as many inputs or outputs as they want and feed the simulated system real time data using the interactive debugger, a time response series will be plotted in the Time Plot.

5.3 Defrost Fuzzy Controller Result and Analysis

The Defrost fuzzy controller will take two inputs from humidity sensor and door open detecting sensor. When door opening is detected, a timer will start timing the length of the door opening. If the length of time is not long enough to start a defrost cycle, then the time will be accumulated until the defrost cycle kicks in. Then the timer will be reset to zero. Since the fuzzy defrost system only defrosts when necessary, the length of the defrosting cycle is reduced. This helps to solve the problem the automatic defrost system has. The automatic defrost system has a fixed length of defrosting cycle1 hour) no matter how much frost is. Testing is done using fuzzyTech's interactive analyzers. First from the 3D transfer space (Figure 5.3), we can see the fuzzy controller has a wider control zone with emphasis on some regions. This is largely due to the non-linearity of the door open activity and different humidity situations. With more information being considered, a wiser decision could be made by using fuzzy controller. In the plot the deep color and shallow color represent extreme situations and these two regions do not account for the main part of the control space. There are more areas representing the situation between the two extreme points, which is reasonable, and therefore the consequent output action will be necessary and waste no extra energy. Moreover, the output will correspond differently to different input values. For each one of the infinite inputs on the universe discourse, the output will have one and only one unique value for it. This is the biggest achievement over using conventional controller or look-up table. It is almost impossible for the conventional design to take all situations into consideration while still using the

low cost micro-controllers. The code size of the defrosting fuzzy controller is only less than 6K.

In the time plot, real time data were fed through the interactive debugger. The time plot is shown in Figure 5.2. The data fed are randomly picked within the universe discourse of one input with the other unchanged. Output is observed and plotted with both inputs in the same plot. We noticed that the output – length of defrost cycle acts to the opposite of the trend of humidity and accumulated door open time. As humidity increases the period between two consecutive defrost cycles is shortened, when the humidity decrease then the defrosting cycle is prolonged. When there are frequent door opening activities, defrosting cycle is shorter than normal. And the defrosting cycle is prolonged if there are fewer doors opening activity. All these agree our intuitive.

The concept and structure of the designed controller has shown its superior to the original automatic defrost system. This adaptive defrost fuzzy controller has its flexibility in dealing with the entire situation the refrigerator might have and not adding too much overhead. And due to the prompt response of the controller to different situations, no pre-fixed schedule and extra defrosting length are needed. Therefore, energy is saved. However, the result is far from optimal. This is partially due to the facts that for each different individual refrigerator, the temperature characteristic is different. Further membership and rule tuning are required for each individual type of refrigerator. A thorough characterization of the refrigerator's temperature behavior will be needed. According to the results of characterization, the universe discourse, label scope and the shape of the membership might have to be modified base on the new information.

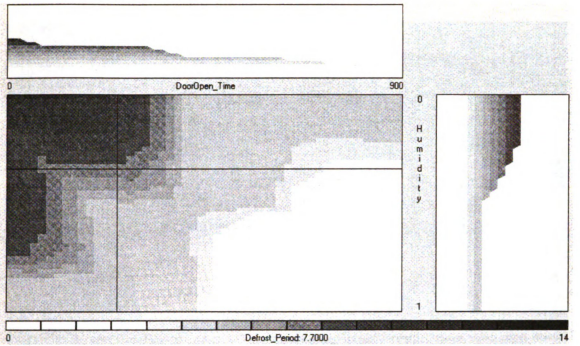


Figure 5.1: Transfer Plot of Fuzzy Defrosting Controller

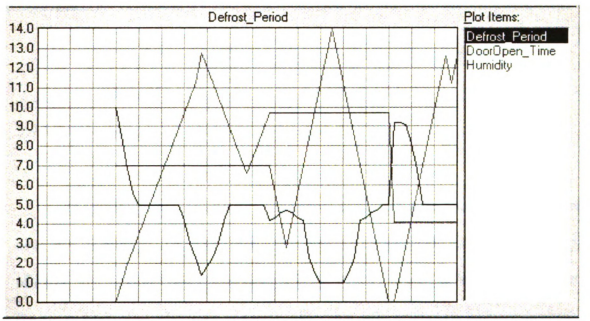


Figure 5.2: Time Response of Fuzzy Defrosting Controller

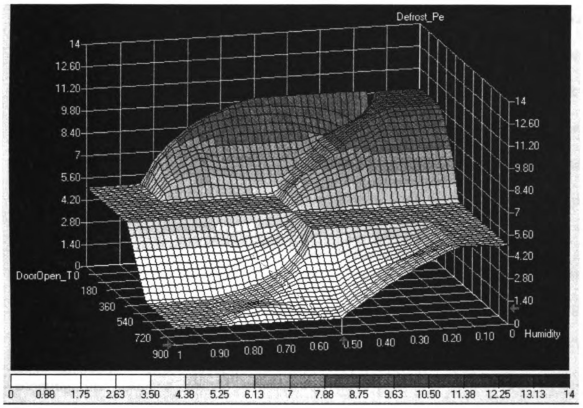


Figure 5.3: 3D Plot of Fuzzy Defrosting Controller

5.4 **Conclusions**

The fuzzy logic concepts and design methodologies were presented in this thesis over several chapters. The work in this thesis serves to help understand the practical use of fuzzy logic in embedded system design. These topics included defining the structure of the fuzzy controller, choosing of the fuzzification, rule evaluation and defuzzification algorithms, and designing hardware/software and analysis.

From the analysis, the advantage of fuzzy logic design is evident. As it is shown in Figure 5.3, fuzzy controller deals with more situations with less system resources. The

whole code for programming the fuzzy controller is less than 6K, which is small enough to be stored in the 32K on-board memory of some embedded system controller, i.e., handyboard. Since for each combination of the inputs there is an unique output that can be easily found through the membership function, real time processing – the bottleneck in conventional controller for computing a complicated model – will never going to be a problem. The quickness and easiness to get an output for a fuzzy controller is almost like a lookup table. But a lookup table can never hold a large amount of data without requiring more memory, and be able to store infinite combinations of input states.

Moreover, fuzzy controller is better off solving ill-defined, non-linear problems and helps to solve problems with conflicting goals. In this fuzzy defrosting controller case, the two conflicting goals energy efficiency and no frost results are well resolved by adopting the adaptive defrost mechanism. Combining the simple but optimal mathematical model – i.e., membership function – and heuristic estimations – i.e., the average door open time is 30 to 90 seconds, no prefixed defrost schedule is needed. Thereafter, the length of each defrost cycle will be greatly shortened. The energy save from the improved fuzzy defrost system is expected to be 50% while the no frost result can still be guaranteed.

At last, since fuzzy logic controllers rely on the values within the universe discourse and label scope to represent the numerical characteristic of the problem, membership function and rule tuning are required for the same system working in different environments. However, this very reason makes the fuzzy logic design methodology very attractive and provides designers with a flexible but powerful design tool which is easier to be implemented and optimized.

Chapter 6

RECOMMENDATIONS FOR FUTURE WORK

The demonstration of using fuzzyTech software to design the defrost system were provided in the previous chapter and a recommended fuzzy controller for setting temperature for energy saving and user- friendly interface was proposed. The next step would be to implement the defrost controller onto the refrigerator, modify and tune the membership function and rule base of the fuzzy controller based on the characterization of the refrigerator. In addition to this, a software implementation for the fuzzy controller that automatically set the temperature is recommended. This would provide the opportunity to further explore the concepts and methodologies of fuzzy design and the necessary experience to develop the fuzzy controller.

Appendix A

DESIGN OF FUZZY DEFROSTING SYSTEM

A.1 Introduction

This is the fully documented design of adaptive defrosting system using fuzzy logic. In this part, the general information of the design, the fuzzy defrosting system structure and the definition of the linguistic variables for the fuzzy defrosting controller are described. Details on input/output interface, definition of rule block, and the methods that are used for fuzzy logic computations are presented.

A.2 General Information

Author:	Weijjiang Robert Yu
Created:	11/17/98
Print Date:	5/22/99

Edition

Edition Name:	fuzzyTECH 5.01 MCU-HC11/12 Explorer
Neuro Modul:	NeuroFuzzy add-on Module not installed

A.3 Project Description

First Edition.

Linguistic Input Variables	2
Linguistic Output Variables	1
Intermediate Variables	0
Rule Blocks	1
Rules	15
Membership Functions	11

TableA-1: Project Statistics

A.4 System Structure

The system structure, as shown in Figure A-1, identifies the fuzzy logic inference flow from the input variables to the output variables. The fuzzification in the input interfaces translates analog inputs into fuzzy values. The fuzzy inference takes place in rule blocks that contain the linguistic control rules. The outputs of these rule blocks are linguistic variables. The defuzzification in the output interfaces translates them into analog variables.

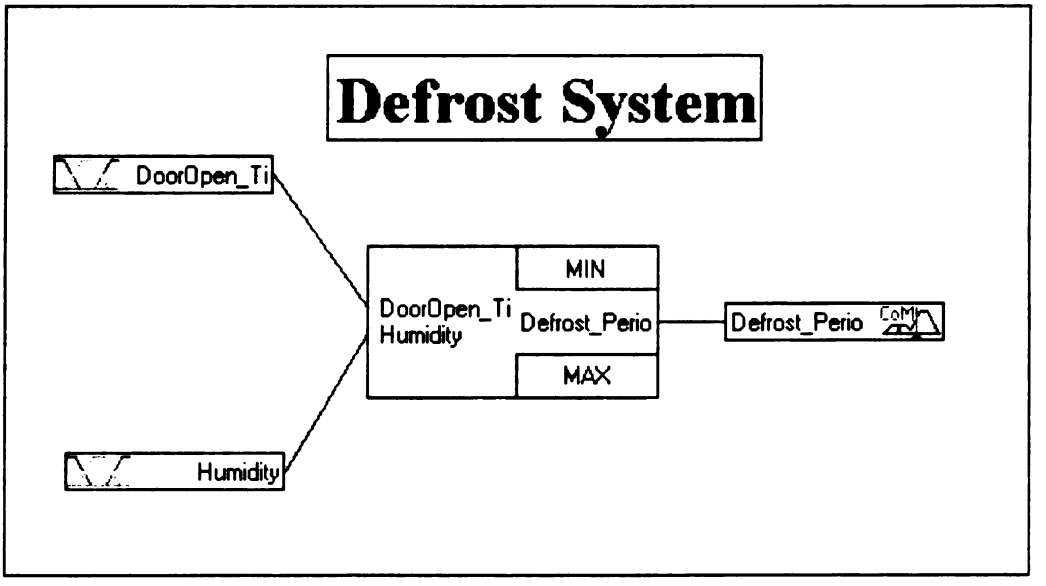


Figure A-1: Structure of the Fuzzy Logic Defrosting System

Figure A-1 shows the whole structure of this fuzzy system including input interfaces, rule blocks and output interfaces. The connecting lines symbolize the data flow.

A.5 Linguistic Variables

This appendix contains the definition of all linguistic variables and of all membership functions. Linguistic variables are used to translate real values into linguistic values. The possible values of a linguistic variable are not numbers but so called 'linguistic terms'. For example:

To translate the real variable 'Temperature' into a linguistic variable three terms, 'Cold', 'Pleasant' and 'Warm' are defined. Depending on the current temperature level each of these terms describes the 'temperature' more or less well. Each term is defined by a membership function (MBF). Each membership function defines for any value of the input variable the associated degree of membership of the linguistic term. The membership functions of all terms of one linguistic variable are normally displayed in one graph. The following figure (Figure A-2) plots the membership functions of the three terms for the example 'Temperature'.

A 'Temperature' of 66 °F is a member of the MBFs for the terms:

Cold to the degree of 0.8

Pleasant to the degree of 0.2

Warm to the degree of 0.0

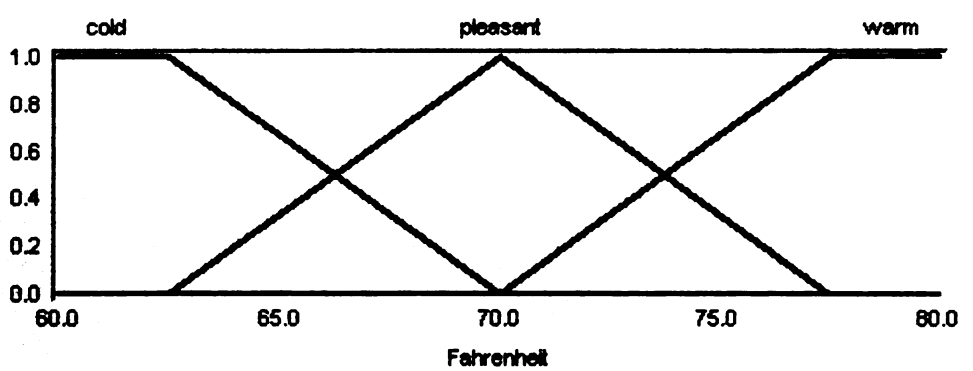


Figure A-2: Membership Function of 'Temperature'

Linguistic variables have to be defined for all input, output and intermediate variables. The membership functions are defined using a few definition points only. The following table (Table A-2) lists all linguistic variables of the system, their types and their term names.

Variable Name	Type	Term Names
DoorOpen_Time	Input	Extreme_Short, Short, Normal, Long, Extreme_Long
Humidity	Input	Dry, Normal, Wet
Defrost_Period	Output	Short, Normal, Long

Table A-2: Linguistic Variables

The properties of all input variables are listed in Table A-3.

Input Variable	Min	Max	Base Var. Unit	Fuzzification
DoorOpen_Time	0	900	Second	Compute MBF
Humidity	0	1	Percent	Compute MBF

Table A-3: Properties of Linguistic Input Variables

The properties of all output variables are listed in the following table (Table A-4). The default value of a variable is used if no rule is firing for this variable. Different methods can be used for the defuzzification, resulting either into the **Most Plausible Result** or the **Best Compromise**.

The **Best Compromise** is produced by the methods:

- CoM (Center of Maximum)
- CoA (Center of Area)
- CoA BSUM, a version especially for efficient VLSI implementations

The **Most Plausible Result** is produced by the methods:

- MoM (Mean of Maximum)
- MoM BSUM, a version especially for efficient VLSI implementations

Output Variable	Min	Default	Max	Base Var. Unit	Defuzzification
Defrost_Period	0	-	14	Days	CoM

Table A-4: Properties of Linguistic Output Variables

A.5.1 Output variable "Defrost_Period"

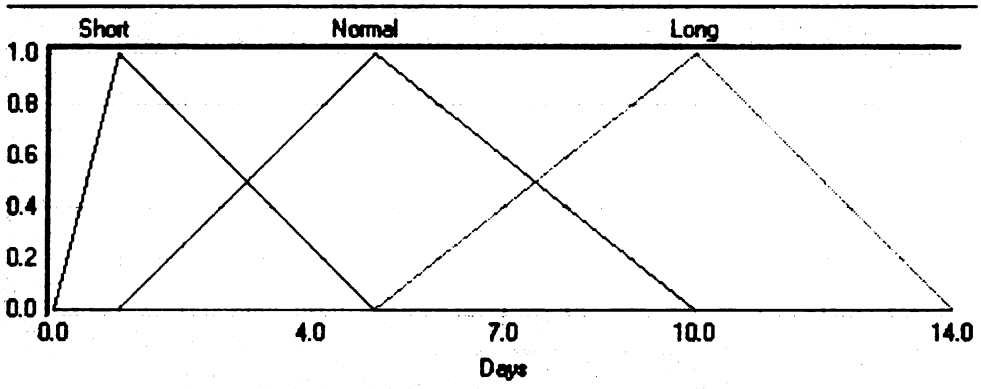


Figure A-3: Membership Function of "Defrost_Period"

The output membership function is defined as shown in Figure A-3.

Term Name	Shape/Par.	Definition Points (x, y)		
Short	linear	(0, 0) (14, 0)	(1, 1)	(5, 0)
Normal	linear	(0, 0) (10, 0)	(1, 0) (14, 0)	(5, 1)
Long	linear	(0, 0) (14, 0)	(5, 0)	(10, 1)

Table A-5: Definition Points of Membership Function for "Defrost_Period"

A.5.2 Input variable "DoorOpen_Time"

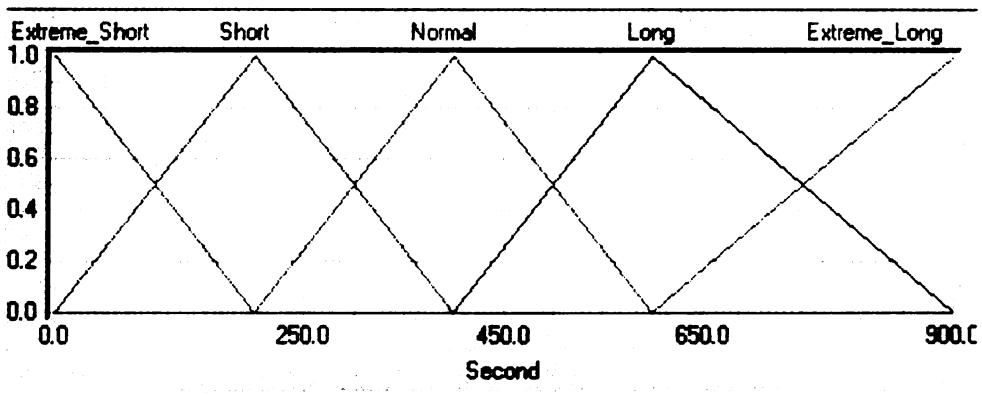


Figure A-4: Membership Function of "DoorOpen_Time"

Term Name	Shape/Par.	Definition Points (x, y)		
Extreme_Short	linear	(0, 1)	(200, 0)	(900, 0)
Short	linear	(0, 0) (900, 0)	(200, 1)	(400, 0)
Normal	linear	(0, 0) (600, 0)	(200, 0) (900, 0)	(400, 1)
Long	linear	(0, 0) (900, 0)	(400, 0)	(600, 1)
Extreme_Long	linear	(0, 0)	(600, 0)	(900, 1)

Table A-6: Definition Points of Membership Functions for "DoorOpen_Time"

Overall time of open door between two consective defrost cycle is defined in Table A-6 and the membership function is shown in Figure A-4.

A.5.3 Input variable "Humidity"

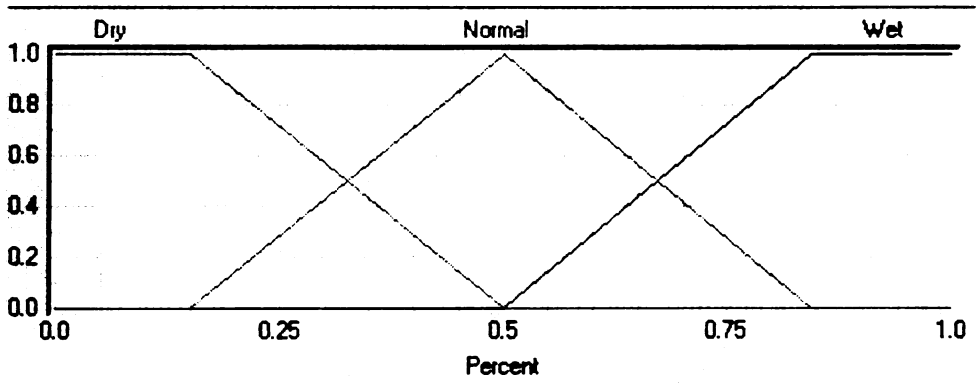


Figure A-5: Membership Function of "Humidity"

Term Name	Shape/Par.	Definition Points (x, y)		
Dry	linear	(0, 1) (1, 0)	(0.15, 1)	(0.5, 0)
Normal	linear	(0, 0) (0.845, 0)	(0.15, 0) (1, 0)	(0.5, 1)
Wet	linear	(0, 0) (1, 1)	(0.5, 0)	(0.845, 1)

Table A-7: Definition Points of Membership Functions for "Humidity"

The membership function of "Humidity" is shown in Figure A-5 and defined as indicated in Table A-7.

A.6 Interfaces

Input Variable	IO-Mapping	Fuzzification
DoorOpen_Time		Compute MBF
Humidity		Compute MBF

Table A-8: Input Interfaces

Output Variable	IO-Mapping	Defuzzification
Defrost_Period		CoM

Table A-9: Output Interfaces

The methods that will be used for fuzzy logic computations for fuzzification and defuzzification are shown in Table A-8 and Table A-9.

A.7 Rule Blocks

The rule blocks contain the control strategy of a fuzzy logic system. Each rule block confines all rules for the same context. A context is defined by the same input and output variables of the rules.

The rules' 'if' part describes the situation, for which the rules are designed. The 'then' part describes the response of the fuzzy system in this situation. The degree of support (DoS) is used to weigh each rule according to its importance.

The processing of the rules starts with calculating the 'if' part. The operator type of the rule block determines which method is used. The operator types MIN-MAX, MIN-AVG and GAMMA are available. The characteristic of each operator type is influenced by an additional parameter. For example:

MIN-MAX, parameter value 0 = Minimum Operator (MIN)

MIN-MAX, parameter value 1 = Maximum Operator (MAX)

GAMMA, parameter value 0 = Product Operator (PROD)

The minimum operator is a generalization of the Boolean 'and'; the maximum operator is a generalization of the Boolean 'or'.

The fuzzy composition eventually combines the different rules to one conclusion. If the BSUM method is used all firing rules are evaluated, if the MAX method is used only the dominant rules are evaluated.

A.7.1 Rule block "RB1"

Parameter

Aggregation:	MINMAX
Parameter:	0.00
Result Aggregation:	MAX
Number of Inputs:	2
Number of Outputs:	1
Number of Rules:	15

Table A-10 defines all the rules for fuzzy defrosting controller.

IF		THEN	
DoorOpen_Time	**Humidity**	**DoS**	**Defrost_Period**
Extreme_Short	Dry	0.00	Long
Extreme_Short	Normal	0.56	Long
Extreme_Short	Wet	0.20	Normal
Short	Dry	0.81	Long
Short	Normal	0.59	Normal
Short	Wet	0.48	Normal
Normal	Dry	0.35	Normal
Normal	Normal	0.90	Normal
Normal	Wet	0.82	Normal
Long	Dry	0.75	Normal
Long	Normal	0.17	Normal
Long	Wet	0.86	Short
Extreme_Long	Dry	0.71	Normal
Extreme_Long	Normal	0.52	Short
Extreme_Long	Wet	0.30	Short

Table A-10: Rules of Rule Block "RB1"

Appendix B

FUZZY TECHNOLOGY LANGUAGE FOR FUZZY DEFROSTING SYSTEM

B.1 Introduction

Appendix B is the design of the fuzzy defrosting system written in text format. As we mentioned earlier, this is also a kind of software code that can be used to implement the fuzzy controller. The difference between this fuzzy technology language and the other popular programming languages is that the fuzzy technology language has to be precompiled using either compilers for C or for assembly language to generate real C codes or assembly codes before use.

B.2 Fuzzy Technology Language for Fuzzy Defrosting System

```
PROJECT {
        NAME = DEFROST_SYSTEM.FTL;
        TITLE =DEFROST_SYSTEM;
        AUTHOR = Weijjiang Robert Yu;
        DATEFORMAT = M.D.YY;
        LASTCHANGE = 11.18.98;
        CREATED    = 11.17.98;
        SHELL = EXPL12;
        COMMENT {
First Edition.} /* COMMENT */
 SHELLOPTIONS {
        ONLINE_REFRESHTIME = 55;
        ONLINE_TIMEOUTCOUNT = 1100;
        ONLINE_PORT = SERIAL;
        ONLINE_CODE = OFF;
        COMMENTS    = ON;
        TRACE_BUFFER = (ON, PAR(100));
```

```
      FTL_BUFFER = (OFF, PAR(1));
      PASSWORD  = OFF;
      PUBLIC_IO = ON;
      FAST_CMBF = OFF;
      FAST_COA  = OFF;
      FILE_CODE = OFF;
      BTYPE = 8_BIT;
} /* SHELLOPTIONS */
  MODEL {
     VARIABLE_SECTION {
        LVAR {
            NAME   = Defrost_Period;
            BASEVAR = Days;
            LVRANGE = MIN(0.000000), MAX(14.000000),
            MINDEF(0), MAXDEF(255),
          DEFAULT_OUTPUT(0.500000);
           RESOLUTION = XGRID(0.100000), YGRID(1.000000),
                        SHOWGRID (ON), SNAPTOGRID(ON);
COMMENT {
The time between two consecutive defrost cycles.}  /* COMMENT */
          COLOR = RED (0), GREEN (0), BLUE (255);
    TERM {
            TERMNAME = Short;
            POINTS = (0.000000, 0.000000),
                     (1.000000, 1.000000),
                     (5.000000, 0.000000),
                     (14.000000, 0.000000);
            SHAPE = LINEAR;
            COLOR = RED (0), GREEN (0), BLUE (255);
    }
          TERM {
                   TERMNAME = Normal;
                   POINTS = (0.000000, 0.000000),
                            (1.000000, 0.000000),
                            (5.000000, 1.000000),
                            (10.000000, 0.000000),
                            (14.000000, 0.000000);
                   SHAPE = LINEAR;
                   COLOR = RED (128), GREEN (0), BLUE (0);
          }
          TERM {
                   TERMNAME = Long;
                   POINTS = (0.000000, 0.000000),
                            (5.000000, 0.000000),
                            (10.000000, 1.000000),
                            (14.000000, 0.000000);
```

```
                    SHAPE = LINEAR;
                    COLOR = RED (0), GREEN (128), BLUE (128);
    }
        } /* LVAR */
        LVAR {
                    NAME   = DoorOpen_Time;
                    BASEVAR = Second;
                    LVRANGE = MIN(0.000000), MAX(900.000000),
                    MINDEF(0), MAXDEF(255),
                    DEFAULT_OUTPUT(0.500000);
                    RESOLUTION = XGRID(5.000000), YGRID(1.000000),
                    SHOWGRID (ON), SNAPTOGRID(ON);
 COMMENT {
Overall time of open door between two consective defrost cycle.} /* COMMENT */
            COLOR = RED (0), GREEN (128), BLUE (0);
        TERM {
                    TERMNAME = Extreme_Short;
                    POINTS = (0.000000, 1.000000),
                              (200.000000, 0.000000),
                              (900.000000, 0.000000);
                    SHAPE = LINEAR;
                    COLOR = RED (255), GREEN (0), BLUE (0);
        }
        TERM {
                    TERMNAME = Short;
                    POINTS = (0.000000, 0.000000),
                              (200.000000, 1.000000),
                              (400.000000, 0.000000),
                              (900.000000, 0.000000);
                    SHAPE = LINEAR;
                    COLOR = RED (0), GREEN (128), BLUE (0);
    }
        TERM {
                    TERMNAME = Normal;
                    POINTS = (0.000000, 0.000000),
                              (200.000000, 0.000000),
                              (400.000000, 1.000000),
                              (600.000000, 0.000000),
                            (900.000000, 0.000000);
                    SHAPE = LINEAR;
                    COLOR = RED (0), GREEN (128), BLUE (128);
    }
        TERM {
                    TERMNAME = Long;
                    POINTS = (0.000000, 0.000000),
                              (400.000000, 0.000000),
```

```
                    (600.000000, 1.000000),
                    (900.000000, 0.000000);
                SHAPE = LINEAR;
                COLOR = RED (128), GREEN (0), BLUE (0);
    }
    TERM {
                TERMNAME = Extreme_Long;
                POINTS = (0.000000, 0.000000),
                    (600.000000, 0.000000),
                    (900.000000, 1.000000);
                SHAPE = LINEAR;
                COLOR = RED (0), GREEN (128), BLUE (128);
    }
} /* LVAR */
    LVAR {
                NAME    = Humidity;
                BASEVAR = Percent;
                LVRANGE = MIN(0.000000), MAX(1.000000),
                MINDEF(0), MAXDEF(255),
                DEFAULT_OUTPUT(0.500000);
                RESOLUTION = XGRID(0.005000), YGRID(1.000000),
                SHOWGRID (ON), SNAPTOGRID(ON);
                COLOR = RED (255), GREEN (0), BLUE (0);
        TERM {
                TERMNAME = Dry;
                POINTS = (0.000000, 1.000000),
                    (0.150000, 1.000000),
                    (0.500000, 0.000000),
                    (1.000000, 0.000000);
                SHAPE = LINEAR;
                COLOR = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
                TERMNAME = Normal;
                POINTS = (0.000000, 0.000000),
                    (0.150000, 0.000000),
                    (0.500000, 1.000000),
                    (0.845000, 0.000000),
                    (1.000000, 0.000000);
                SHAPE = LINEAR;
                COLOR = RED (0), GREEN (128), BLUE (0);
    }
        TERM {
                TERMNAME = Wet;
                POINTS = (0.000000, 0.000000),
                    (0.500000, 0.000000),
```

```
                    (0.845000, 1.000000),
                    (1.000000, 1.000000);
                SHAPE = LINEAR;
                COLOR = RED (0), GREEN (0), BLUE (255);
            }
        } /* LVAR */
} /* VARIABLE_SECTION */

    OBJECT_SECTION {
            INTERFACE {
            INPUT = (Humidity, CMBF);
            POS = -213, -1;
    }
    INTERFACE {
            INPUT = (DoorOpen_Time, CMBF);
            POS = -219, -146;
    }
        INTERFACE {
                OUTPUT = (Defrost_Period, COM);
                POS = 109, -74;
    }
        RULEBLOCK {
                NAME = RB1;
                INPUT = DoorOpen_Time, Humidity;
                OUTPUT = Defrost_Period;
                AGGREGATION = (MIN_MAX, PAR (0.000000));
                RESULT_AGGR = MAX;
                POS = -66, -102;
          RULES {
                IF   DoorOpen_Time = Extreme_Short
                     AND Humidity = Dry
                THEN Defrost_Period = Long   WITH 0.000;
                IF   DoorOpen_Time = Extreme_Short
                     AND Humidity = Normal
                THEN Defrost_Period = Long   WITH 0.563;
                IF   DoorOpen_Time = Extreme_Short
                     AND Humidity = Wet
                THEN Defrost_Period = Normal   WITH 0.195;
                IF   DoorOpen_Time = Short
                     AND Humidity = Dry
                THEN Defrost_Period = Long   WITH 0.813;
                IF   DoorOpen_Time = Short
                AND Humidity = Normal
            THEN Defrost_Period = Normal   WITH 0.586;
            IF   DoorOpen_Time = Short
```

```
                    AND Humidity = Wet
                THEN  Defrost_Period = Normal   WITH 0.477;
                IF    DoorOpen_Time = Normal
                    AND Humidity = Dry
                THEN  Defrost_Period = Normal   WITH 0.352;
                IF    DoorOpen_Time = Normal
                    AND Humidity = Normal
                THEN  Defrost_Period = Normal   WITH 0.898;
                IF    DoorOpen_Time = Normal
                    AND Humidity = Wet
                THEN  Defrost_Period = Normal   WITH 0.820;
                IF    DoorOpen_Time = Long
                    AND Humidity = Dry
                 THEN  Defrost_Period = Normal   WITH 0.750;
                 IF    DoorOpen_Time = Long
                    AND Humidity = Normal
                 THEN  Defrost_Period = Normal   WITH 0.172;
                 IF    DoorOpen_Time = Long
                    AND Humidity = Wet
                 THEN  Defrost_Period = Short   WITH 0.859;
                 IF    DoorOpen_Time = Extreme_Long
                    AND Humidity = Dry
                 THEN  Defrost_Period = Normal   WITH 0.711;
                 IF    DoorOpen_Time = Extreme_Long
                    AND Humidity = Normal
                  THEN  Defrost_Period = Short   WITH 0.516;
                  IF    DoorOpen_Time = Extreme_Long
                    AND Humidity = Wet
                  THEN  Defrost_Period = Short   WITH 0.305;
            } /* RULES */
    }
        REMARK {
                    TEXT = Defrost System;
                    POS = -86, -195;
                    FONTSPEC = -32, 700, 0, 0, 0, 18, 0;
                    FONTNAME =Times New Roman;
                    COLOR = RED (0), GREEN (0), BLUE (128);
            }
        } /* OBJECT_SECTION */
    } /* MODEL */
} /* PROJECT */

TERMINAL {
        BAUDRATE    = 19200;
        DATABITS    = 8;
        PARITY      = 0;
```

```
        STOPBITS    = 1;
        PROTOCOL    = NO;
        CONNECTION  = PORT1;
        INPUTBUFFER  = 256;
        OUTPUTBUFFER = 256;
} /* TERMINAL */

ONLINE {
        TIMESTAMP = 1998111821523500;
} /* ONLINE */
```

Appendix C

ASSEMBLY CODE FOR FUZZY DEFROSTING CONTROLLER

C.1 Introduction

This appendix is documented with the assembly code generated for the fuzzy defrosting controller for using Motorola's MC68HC11/MC68HC12 micro-controller. The code is ready to be downloaded to the micro- controller if the current design is not to be changed. Any changes of parameters to the design will need re-generation of the code.

C.2 Code

```
;----- FuzzyTECH 5.01 MCU-HC11/12 Explorer --- Pre-assembler
;---------- code generation date: Sat May 22 15:14:38 1999 -------
;------------------------ project: DEFROS1 ----------------------------
;------------------------------- MCU: 12 -------------------------------
          XDEF   _defros1
          XDEF   _initdefros1
;-------------------- input/output interface of controller ------------
          XDEF   _DoorOpen_Time_defros1    ;00H .. FFH
          XDEF   _Humidity_defros1         ;00H .. FFH
          XDEF   _Defrost_Period_defros1   ;00H .. FFH
;-------------------- external functions of fuzzy library -------------
          XREF   flms
          XREF   com
;-------------------- external variables of fuzzy library -------------
          XREF   fuzptr
          XREF   _invalidflags
          XREF   itcnt
          XREF   tpptr
          XREF   crisp
          XREF   otcnt
;---------------------------------------------------------------------

                              switch        .bss
```

```
;------------------------------ RAM i/o-vars -----------------------
fuzvals:                                ;8 + 3 + 0
fvs:
_t_DoorOpen_Time_defros1:  ds.b   5
_t_Humidity_defros1:       ds.b   3
_t_Defrost_Period_defros1: ds.b   3
crispio:                                ;3
_DoorOpen_Time_defros1:    ds.b   1
_Humidity_defros1:         ds.b   1
_Defrost_Period_defros1:   ds.b   1
switch        .text
;----------------- standard term definition (x1, x2, x3, x4) ----------
tpts:
            dc.b   $00, $00, $00, $39
            dc.b   $00, $39, $39, $71
            dc.b   $39, $71, $71, $AA
            dc.b   $71, $AA, $AA, $FF
            dc.b   $AA, $FF, $FF, $FF
            dc.b   $00, $00, $26, $80
            dc.b   $26, $80, $80, $D7
            dc.b   $80, $D7, $FF, $FF
;----------------------- xcom table (defuzzification) ----------------
xcom:
            dc.b   $12, $5B, $B6
;------------------------------ rule table(s) ----------------------
_rt0:
            dc.w   fvs+$0, fvs+$5, $FFFE
            dc.w   fvs+$A, $FFFE
            dc.w   fvs+$0, fvs+$6, $FFFE
            dc.w   fvs+$A, $FFFE
            dc.w   fvs+$0, fvs+$7, $FFFE
            dc.w   fvs+$9, $FFFE
            dc.w   fvs+$1, fvs+$5, $FFFE
            dc.w   fvs+$A, $FFFE
            dc.w   fvs+$1, fvs+$6, $FFFE
            dc.w   fvs+$9, $FFFE
            dc.w   fvs+$1, fvs+$7, $FFFE
            dc.w   fvs+$9, $FFFE
            dc.w   fvs+$2, fvs+$5, $FFFE
            dc.w   fvs+$9, $FFFE
            dc.w   fvs+$2, fvs+$6, $FFFE
            dc.w   fvs+$9, $FFFE
            dc.w   fvs+$2, fvs+$7, $FFFE
            dc.w   fvs+$9, $FFFE
            dc.w   fvs+$3, fvs+$5, $FFFE
            dc.w   fvs+$9, $FFFE
```

```
        dc.w   fvs+$3, fvs+$6, $FFFE
        dc.w   fvs+$9, $FFFE
        dc.w   fvs+$3, fvs+$7, $FFFE
        dc.w   fvs+$8, $FFFE
        dc.w   fvs+$4, fvs+$5, $FFFE
        dc.w   fvs+$9, $FFFE
        dc.w   fvs+$4, fvs+$6, $FFFE
        dc.w   fvs+$8, $FFFE
        dc.w   fvs+$4, fvs+$7, $FFFE
        dc.w   fvs+$8, $FFFF

rt0:
        dc.b   $00, $8F, $31, $CF, $95, $79, $59, $E5, $D1, $BF, $2B
        dc.b   $DB, $B5, $83, $4D
;---------------------------------------------------------------------
_defros1:
;------------------------------ fuzzification ----------------------
        ldy    #fuzvals
        ldd    #tpts
        std    tpptr
        ldab   #$5
        stab   itcnt
        ldab   _DoorOpen_Time_defros1
        stab   crisp
        jsr    flms

        ldab   #$3
        stab   itcnt
        ldab   _Humidity_defros1
        stab   crisp
        jsr    flms
;------------------------------ inference init ---------------------
;------------------------------ rule block 0 -----------------------
        ldx    #_rt0
        ldy    #rt0
        ldaa   #255
        sec
        sei                     ;patch
        revw                      ;min aggregation, product composition
        cli                     ;patch
;------------------------------ defuzzification --------------------
        clr    _invalidflags
        ldy    #fuzvals + $8

        ldx    #xcom + $0
        ldab   #$3
```

```
            stab   otcnt
            jsr    com
            bcc    out0valid
            ldab   #$09
out0valid:     stab   _Defrost_Period_defros1

            ldab   _invalidflags
            rts                         ;end of fuzzy controller
;-------------------------------- initdefros1 -------------------------
_initdefros1:
            ldy    #$3
            ldaa   #0
begin:
            staa   fuzvals + $7,y
            dey
            bne    begin
            rts

;data size knowledge base (bytes):
;RAM:       14   0000EH
;ROM:      140   0008CH
;TOTAL:    154   0009AH
;
            end
```

BIBLIOGRAPHY

Bibliography

[1] Zadeh, L. A., "Fuzzy Sets," Information and Control, Vol. 8, 1965
[2] "Fuzzy Logic 2.0", Motorola
[3] James F. Brule, "FUZZY SYSTEMS - A TUTORIAL", [Online] Available http://www.austinlinks.com/Fuzzy/tutorial.html
[4] S. Korner, "Laws of thought," Encyclopedia of Philosophy, Vol. 4, MacMillan, NY: 1967, pp. 414-417.
[5] C. Lejewski, "Jan Lukasiewicz," Encyclopedia of Philosophy, Vol. 5, MacMillan, NY: 1967, pp. 104-107.
[6] Witold Pedrycz, "Fuzzy Control and Fuzzy Systems", second, extended edition, 1993
[7] George J. Klir/Bo Yuan, "Fuzzy Sets and Fuzzy Logic -- Theory and Applications"
[8] Pedrycz, W., 1990. "Fuzzy sets framework for development of perception perspective," Fuzzy Sets & Sys.
[9] Pedrycz, W., 1992. "Selected issues of frame of knowledge representation realized by means of linguistic labels," Int. J. Intelligent Syst.
[10] Constantin Von Altrock, "Fuzzy Logic and NeuroFuzzy Applications Explained",
[11] E.H.Mamdani, "Twenty Years of Fuzzy Control: Experiences Gained and Lessons Learned", IEEE Technology Update Series, Fuzzy Logic Technology and Applications.
[12] "fuzzyTech 4.0 MCU Edition Manual," INFORM/Inform Software Corp.,
[13] Hideyuki Takagi, "Application of Neural Networks and Fuzzy Logic to Consumer Products", IEEE Technology Update Series, Fuzzy Logic Technology and Applications.
[14] [Online] Available http://www.pbs.org/weta/planet/main/software/refrigerator/chill-tech.html
[15] Alan Meier, "Energy Test Procedures for the twenty-first Century", [Online] Available http://eetd.lbl.gov/EA/Buildings/ALAN/Publications/AMCE/AMCE.text.html
[16] James Braham,"Energy Efficiency Sends Appliance Designers into A Spin", Machine Design, August 7, 1997, [Online] Available http://www.penton.com/md/edit/features/80797/appliances080797.html
[17] [Online] Available, Tariq M. Al-Hindi, Amy A. Piotrowski, Carlo Adinata, Joel Ross, Paul Kabir, Aaron Olds, Spring, 1999 http://www.egr.msu.edu/~ee482/Teams/98fall/design1/web/finalreport.htm
[18] [Online] Available http://www.mcu.motsps.com/hc12/home.html
[19] [Online] Available http://www.mcu.motsps.com/lit/hc12.html
[20] "CPU12 Reference Manual", Motorola, [Online] Available http://www.mcu.motsps.com/lit/manuals/cpu12/cpu12rg.pdf
[21] "fuzzyTech 4.0 Online Edition Manual," INFORM/Inform Software Corp.,