2005

An intelligent control system for die casting processes.

Tiebao Yang

University of Windsor

An Intelligent Control System for Die Casting Processes

by

Tiebao Yang

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada
2005

Your file Votre référence
ISBN: 0-494-09792-2
Our file Notre référence
ISBN: 0-494-09792-2

ABSTRACT

The objective of this thesis is to design an intelligent control system for die casting processes involving cooling of a die with multiple channels. The work consists of two parts. First, a correlation between die insert temperature and cooling water outlet temperature is established, which can be used to deduce local die surface temperatures without destructively inserting thermal sensors into a die from its back. Second, a new on-line thermal management scheme is proposed based on an intelligent real-time monitoring and control system (IRMCS) developed for a die insert containing multiple cooling channels. In this scheme, extra cooling water lines controlled by a pump and solenoid valves are hooked up to each established cooling channel. The system is capable of monitoring temperature signals from the die insert and flow rate signals from the cooling lines on the basis of its built-in control algorithms. Pump and solenoid valves can be actuated either automatically or manually to introduce additional cooling water to the die insert for preventing die overheating.

To properly manage thermal distributions in the die, two controllers are implemented separately in the developed system to handle different tasks. One is a local temperature controller that can limit die insert temperature in a given range. Another is a fuzzy controller capable of reducing the temperature of hot spots. Experiments have been carried out on a laboratory die casting process simulator containing a die insert with single cooling channel or multiple cooling lines. The results indicate that the proposed thermal management system is capable of intelligently and continuously managing the water flow rates of multiple cooling lines in a die insert. Hence, the desired thermal pattern of the die becomes achievable.

DEDICATED TO

My wife and her parents

ACKNOWLEDGEMENTS

First and for most, I would like to express my sincere gratitude to my supervisors, Dr. Xiang Chen and Dr. Henry Hu, whose guidance and support made the completion of this project possible. Their insightful thoughts, enthusiasm and confidence have had a great impact on my research. All in one word, I feel lucky to have them as my advisors.

Extra special thanks are due to my committee members, Dr. Huapeng Wu and Dr. Vesslin Stoilov, for their thoughtful questions and valuable advice toward the fulfillment of this project. Special thanks also go to Dr. Xueyuan Nie for serving as the chair of defense.

I am also in debt to Dr. Yeou-li Chu and Mr. Patrick Cheng of Roybi Die Casting (USA) Inc., for sharing their unique experience and knowledge in this research area, and for their endeavor toward our successful on-site demo in their factory.

My thanks also go to Ms. Fang Chen for her previous outstanding work. I also like to thank Mr. Frank Cicchello and Mr. Don Tersigni, the department technicians for their generous help.

The author would like to acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC) and Ryobi Die Casting (USA) for supporting this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

w_{ij}	Element of a matrix or vector in the i th row and j th column
$\mu_A(x)$	General form for member function of the fuzzy set A
ΔU	Smallest detectable voltage
ΔT	Smallest detectable Temperature
ΔF	Smallest detectable Flow Rate
T_d	Die Insert Temperature
T_o	Water Outlet Temperature
$\dot{Q}$	Water Flow Rate
$\dot{Q}_{Ti}$	Total averaged flow rate in i th cooling channel
$e(k)$	Temperature error at time k.
$\Delta e(k)$	Error change at time k.
max(a,b)	Determine the largest value
min(a,b)	Determine the smallest value
T	Temperature
T-max	Upper bound of the desired temperature range
T-min	Lower bound of the desired temperature range
T-target	Maximum temperature that die insert is heated up in the furnace before cooling

Chapter 1

Introduction

This chapter contains a brief introduction to die casting processes, including structure of a casting machine and four stages of a complete casting cycle. Research motivation and literature survey are discussed thereinafter. The outline of this thesis is provided in the last section.

1.1 Die Casting Processes

High pressure die casting is a manufacturing process accomplished by injecting molten metal into a reusable mould, called the die, in which the metal solidifies rapidly into nearly net-shape, smooth-surface components such as engine blocks and transmission cases. A die usually consists of two sections: one is stationary (fixed die half) while another is movable (injector die half) to permit easy removal of the finished castings [1-4]. The schematic view of a die-casting machine is depicted in Figure 1-1. A complete casting cycle can be roughly divided into four stages.

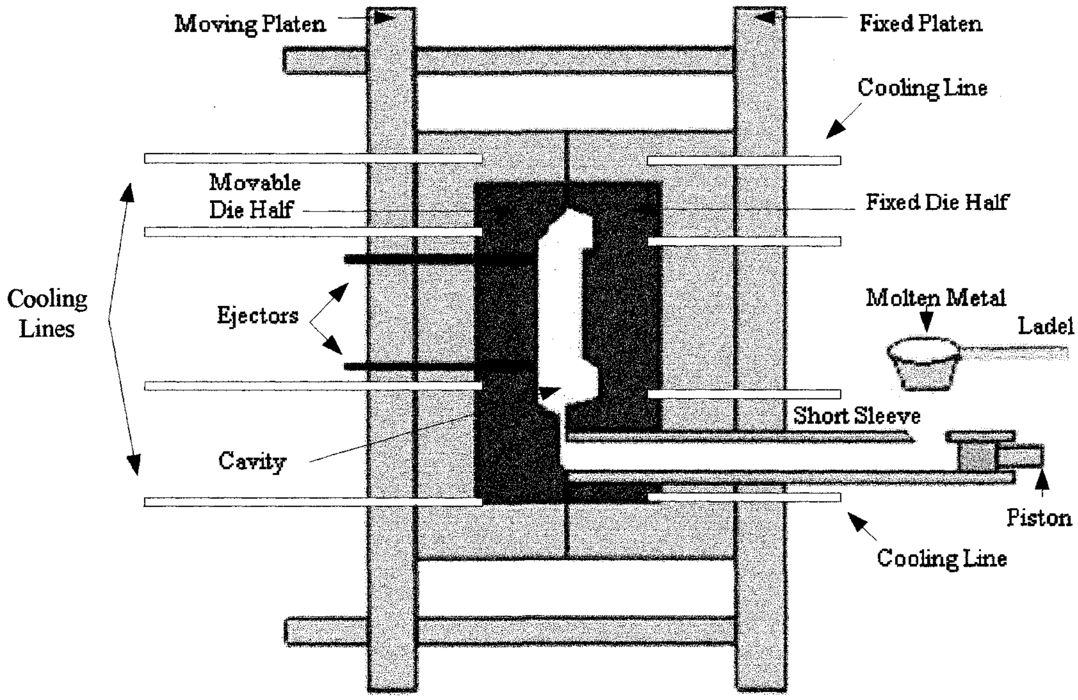


Figure 1.1. Schematic Diagram of a Die Casting Machine.

1. Cavity filling: The two die halves are clamped together tightly by a die casting machine at the start of each casting cycle. Molten alloy is poured into the shot sleeve, and then injected into the die cavity by the piston. The time for cavity filling varies from 5 ms to 150 ms dependent on the casting size.
2. Solidification: The molten metal solidifies in the die cavity under certain applied pressure, usually from 10 to over 30 MPa, to reduce the formation of the porosity and to increase the dimensional accuracy of the part. The die is also chilled by internal cooling water lines to yield high productivity.
3. Casting Ejection: After the molten alloy solidifies, the die halves are separated; the ejectors that are usually driven by hydraulic power expel the casting part, and the part is taken out of platens by a robot or operator.

4. Die Lubrication: The surface of the die is then sprayed by water-based lubricant, and is blown by compressed air in order to further reduce the temperature of the die and oil its surface for next cycle.

Basically there are two functions for a casting die. One is to retain the desired shape of the casting, and another is to remove the heat from the molten metal in a reasonable amount of time. Therefore, the die plays a critical role in removing heat from the molten metal. Proper control of die temperature is essential for producing superior quality components and yielding high production rates.

1.2 Motivation

A die may consist of dozens of die inserts, which are smaller units made of tool steel. Each die insert has at least one internal cooling water line to allow its temperature being controlled properly. Very often, each cooling line is operated with maximum flow rate running through it. The conventional approach to control die temperature is based on the operators' experience and trial-and-error. However, the temperature distribution on a die surface is not homogeneous when controlled by traditional method as illustrated in Figure 1.2, where the picture on right was taken by an infrared camera immediately after a casting part is ejected form a die which produces transmission cases shown by the left picture. The temperature distribution varies significantly from $200^{o}C$ to $400\ ^{o}C$. It is implied that those areas with relatively high temperatures are less efficient at removing heat from the molten metal.

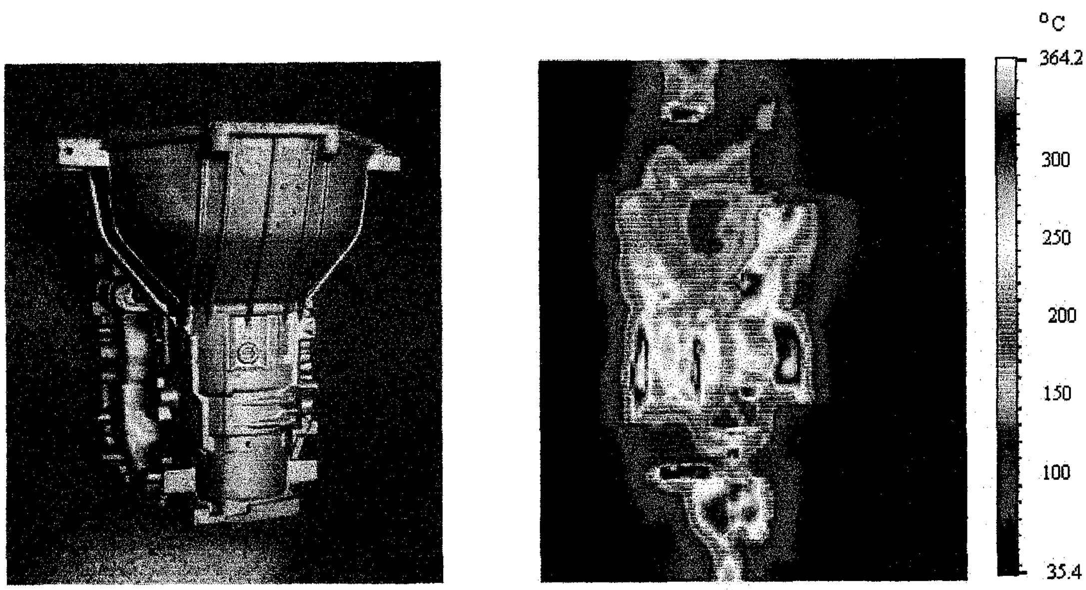


Figure 1.2. Inhomogeneous Temperature Distribution of a Die Surface.

The improper distribution of die temperatures may cause many casting defects. On the one hand, localized over-cooling which leads to the formation of defects such as poor fill can be manually adjusted by decreasing water flow rate in a specific cooling line. On the other hand, casting defects such as porosity and hot tearing, which result from hot spots with abnormal high temperatures present in the die, become difficult to alleviate due to the fact that the established cooling lines are already running at their maximum flow rates.

Owing to the increasing costs of raw materials, workmanship, energy, and the growing competition, die casting manufacturers are motivated to fabricate products with higher quality but lower cost, with greater quantity but smaller scrap rates. The objective of current research is to design and develop an intelligent real-time monitoring and control system (IRMCS) capable of intelligently and continuously managing the water flow rates of cooling lines in a die insert, which makes its desired thermal pattern achievable.

1.3 Literature Survey

In the past, various efforts have been made to develop thermal management systems and a number of commercial temperature control units were designed to help control the die temperature. Some major results are discussed as follows:

On-off Control [5, 6]: Solenoid valves are employed as control actuators to vary the flow rate in cooling water lines, while die temperatures are used as control feedback. Specifically, the valve is fully opened to charge cooling water into a die when die surface temperature reaches a set point, and vice versa. It is obvious that the die temperature tends to oscillate around the set point continuously by this control scheme so that it cannot achieve precise results.

Proportional Control [7, 8]: To overcome the oscillation problem associated with the on-off control, the proportional control was proposed by varying the casting cycle time based on the die surface temperature. It alters the on-off ratio of solenoid valves to control the flow rate of cooling waterlines. The ratio is fixed at 1:1 at a specific temperature set point. Once die temperature is beyond the set point, solenoid valves become fully opened to allow excessive amount of water running through the cooling lines, and vice versa. It can be expected that this approach has a very slow response time and may significantly reduce the productivity.

Proportional-Integral-Differential (PID) Control [9]: Although over 60 years have been past since the first introduction of PID control, the first controller for die thermal management appeared in 1999. This control scheme not only improves the response time but also enhances the control precision. However, due to the fact that the die temperature distribution depends on various die design and process variables for which first principles models are very difficult to be obtained, the three control parameters, proportional, integral and derivative terms, must be tuned individually for particular system through trial-and-error.

So far no paper has been found that exploits advanced control strategy, such as fuzzy algorithm, expert system and etc., to address the problem of die thermal management. It is evident that the existing system lacks the capability of continuously and intelligently controlling the water flow rates in multiple cooling lines in accordance with the fluctuation of local die temperatures, and consequently are not able to correct localized problems which are currently present in most dies used in the industry.

1.1Thesis Outline

The thesis is organized as follows: after the Introduction, Chapter 2 provides some preliminary theories required for the later chapters. In Chapter 3, details of hardware and software structures of a thermal management system are presented. Thermal analysis of a die insert with single cooling channel is covered in Chapter 4. An online control scheme is proposed and implemented in Chapter 5 for a die insert with multiple cooling channels. Finally, the conclusions and suggestions for future work can be found in Chapter 6.

Chapter 2

Preliminary Theories

This chapter is to introduce some important research results that will be applied later in this thesis. In Section 2.1 the method for building a linear model from a given set of measured data points is briefly presented. In Section 2.2, fuzzy set theory is looked over very concisely. The method of designing a fuzzy controller is also illustrated briefly in Section 2.3.

2.1 Linear Model [10]

For a given set of measured data points $\{(x_i, y_i), i = 1,2,...n\}$, where x and y are both scalars, its linear model can be written as

$$y = f(x) = \sum_{j=1}^{m} C_j f_j(x) \tag{2.1}$$

where $f_j(x)$ are known functions, C_j are coefficients to be identified. Define the square error as a cost function

$$E(C_j, j=1,...,m) = \sum_{k=1}^{n}\left[\left[\sum_{j=1}^{m} C_j f_j(x_k)\right] - y_k\right]^2 \tag{2.2}$$

Then, the best values of the unknown C_j that will fit the model can be solved by minimizing the cost function. Take partial derivative of the cost function with respect to C_j, and set

$$\frac{\partial E}{\partial C_j} = 0,\ j = 1,...,m$$

which yield the following normal equations

$$\sum_{j=1}^{m}\left[\sum_{k=1}^{n} f_i(x_k) f_j(x_k)\right] C_j = \sum_{k=1}^{n} f_i(x_k) y_k\ ,\ i = 1,...,m \tag{2.3}$$

The above equation can be written in matrix-vector form as

$$F^T F C = F^T Y \tag{2.4}$$

where superscript T represents matrix transpose, $C = [C_1 \quad C_2 \quad \ldots \quad C_m]$ is the coefficient vector to be determined, $Y = [y_1 \quad y_2 \quad \ldots \quad y_n]$ is defined from the given data, , and F is defined as

$$F = \begin{bmatrix} f_1(x_1) & f_1(x_2) & \cdots & f_1(x_n) \\ f_2(x_1) & f_2(x_2) & \cdots & f_2(x_n) \\ \vdots & \vdots & \vdots & \vdots \\ f_m(x_1) & f_m(x_2) & \cdots & f_m(x_n) \end{bmatrix} \tag{2.5}$$

Assume that $(F^T F)$ is invertible, then the unknown coefficient vector can be solved in the least squares sense

$$C = (F^T F)^{-1} F^T Y \tag{2.6}$$

Thus it is rather straightforward to compute the unknown parameter C.

2.2 Fuzzy Set [11-21]

In classical set theory, an element either belongs or does not belong to a classical set. Its characteristic function can be defined as

$$\chi_A(x) \equiv \begin{cases} 1 & iff \quad x \in A \\ 0 & iff \quad x \notin A \end{cases} \tag{2.7}$$

where A is a classical set. The characteristic function is restricted to the pair {0, 1}, depending on whether x belongs to A or not. The pair is also called the valuation set. The above equation can also be denoted as follows

$$\chi_A(x): \quad X \to \{0,1\} \tag{2.8}$$

where X is universe of discourse, $A \subseteq X$ and $x \in X$. It means that there exists a function $\chi_A(x)$ mapping every element of the set X to the set $\{0, 1\}$. As to the fuzzy set, it is simply generalizing the valuation set from the pair of numbers $\{0, 1\}$ to all numbers found in [0, 1]. A membership function can be defined as

$$\mu_A(x): \quad X \to [0,1] \tag{2.9}$$

It can also be written as a collection of ordered pairs $A = \{(x, \mu_A(x))\}$. Since the interval [0, 1] contains infinite numbers, it is possible for a fuzzy set to have infinite degrees of membership. There are two main characteristics that enable fuzzy systems to have better performance for specific applications. First, fuzzy systems are suitable for uncertain or approximate reasoning, especially for systems of which mathematical models are difficult to derive; second, fuzzy logic allows decision making with estimated values under incomplete or uncertain information. Some important definitions are given as follows.

Definition 1. ($\alpha-cut$) An $\alpha-level$ set of a fuzzy set A of X is a non-fuzzy set denoted by A_α and is defined by

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\} \tag{2.10}$$

where α is a parameter in the range $0 < \alpha \leq 1$.

Definition 2. (Union) Let A and B are fuzzy subsets of a nonempty (crisp) set X, the union of A and B is defined as

$$\mu_{A \cup B} \equiv \mu_A(x) \vee \mu_B(x) = \max[\mu_A(x), \mu_B(x)] \tag{2.11}$$

where $\vee$ is called max operator. The union of two fuzzy sets is related to the logical operation of disjunction (OR) in fuzzy logic.

Definition 3. (Intersection) Let two fuzzy sets A and B are defined over the same universe of discourse X, the intersection of A and B is defined as

$$\mu_{A \cap B} \equiv \mu_A(x) \wedge \mu_B(x) = \min[\mu_A(x), \mu_B(x)] \tag{2.12}$$

where $\wedge$ is min operator. The intersection of two fuzzy sets is related to the logical operation of conjunction (AND) in fuzzy logic.

Definition 4. (Complement) The complement of a fuzzy set A is defined as

$$\mu_{\bar{A}} \equiv 1 - \mu_A(x) \tag{2.13}$$

It is clear that the law of excluded middle and contradiction are not satisfied in fuzzy logic, that is

$$\begin{aligned} A \vee \bar{A} &\neq X \\ A \wedge \bar{A} &\neq \phi \end{aligned} \tag{2.14}$$

Multiple variables in different sets can also have fuzzy relations defined over high-dimensional universe of discourse. Fuzzy algorithms, known as fuzzy if/then rules, are fuzzy relations in linguistic disguise. The fuzzy relations can be defined in terms of two variables as

$$R = \{((x, y), \mu_R(x, y))\} \tag{2.15}$$

where x and y are in different universe of discourse X and Y respectively. In a real problem, x would represent a real variable (such as room temperature) and each set would stand for a linguistic premise (such as room is cold), and y would represent another real variable (such as room pressure) and each set would stand for another linguistic premise (such as room pressure is low). Their resulting fuzzy relation is also a fuzzy set defined by a membership function.

2.3 Fuzzy Control System [13-23]

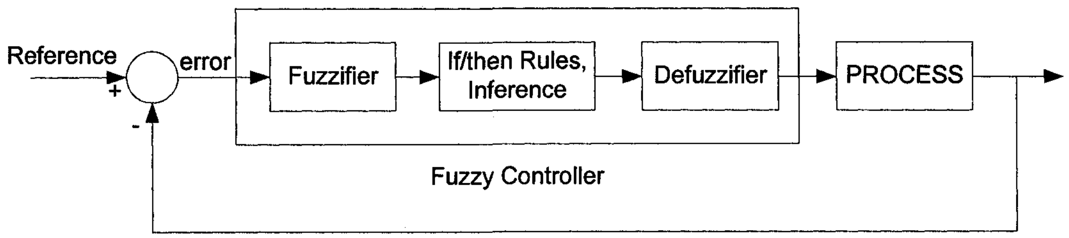


Figure 2.1 Fuzzy Control System.

Although human-like linguistic descriptions are used in fuzzy algorithm to construct a control law, its mathematical foundations involving fuzzy sets and relations are very rigorous. That is also the reason why it can achieve precise control of a process without exact model. The basic structure of a fuzzy controller is outlined in Figure 2.1.

The function of fuzzifier is to compare each error input, which is the difference between the reference input and feedback signal, with a set of possible linguistic variable to determine its membership. Each fuzzy rule has an antecedent statement with an associated fuzzy relation used to calculate the degree of membership. The general form of fuzzy rules can be written as

IF (antecedent) THEN (consequence)

Each consequence, known as control action, is linguistic fuzzy variable. Defuzzifier first creates a combined fuzzy set which is the intersection of output sets for each rule, weighted by the degree of membership for antecedent of the rule, then converts output fuzzy set to a single control value. Perhaps the most frequently used defuzzification method is the centroid or center of area (COA). Its output is defined as

$$u^* = \frac{\sum_{i=1}^{n} u_i \mu_{out}(u_i)}{\sum_{i=1}^{n} \mu_{out}(u_i)} \tag{2.16}$$

where u_i and $\mu_{out}(u_i)$ represent the output of i-th rule and its membership respectively. This method takes into account the area of the resultant membership function $\mu_{out}(u)$ as a whole. When $\mu_{out}(u) = 0$, the crisp output can be simply set to a pre-agreed value, typically $u^* = 0$ in order to avoid dividing by zero. It can be seen that the crisp value u^* is taken to be the geometrical center of the output fuzzy values, which is formed by taking the union of all the contributions of rules whose degree of freedom is bigger than zero.

Three facets must be considered carefully in order to design an effective fuzzy control system. First, linguistic quantities for inputs and outputs need to be decided appropriately. Second, membership function for each linguistic quantity has to be selected properly. Last but not the least, the inference rules ought to be defined accordingly.

Chapter 3

Thermal Management System

This chapter presents the hardware and software structures of a thermal management system that have been built for experiments. A laboratory die casting simulator is also introduced in order to test the performance of the system. A similar yet different design can also be found in [4].

3.1 System Overview

Numerous experiments need to be carried out in order to design and implement the intelligent real-time monitoring and control system (IRMCS) for die casting processes involving cooling of a die with multiple channels. Since it would be very costly to conduct experiments directly on a real die casting machine, a thermal management system for experiments has been built in laboratory. The schematic diagram for a basic configuration is shown in Figure 3.1. Different experiment may need different configuration that will be given in later Chapters. In this thermal management system, personal computer (PC) with a data acquisition board is the command center. A die casting process simulator is employed to preheat the die insert to certain temperature, and then cooling water is applied to cool down the insert. The system monitors the temperature signals that are measured by the thermocouple, and flow rate signals that are measured through the flow sensor. In order to achieve the desired thermal pattern of the die insert, pump and solenoid valve can be actuated through the interface board to continuously adjust the flow rate of the cooling water.

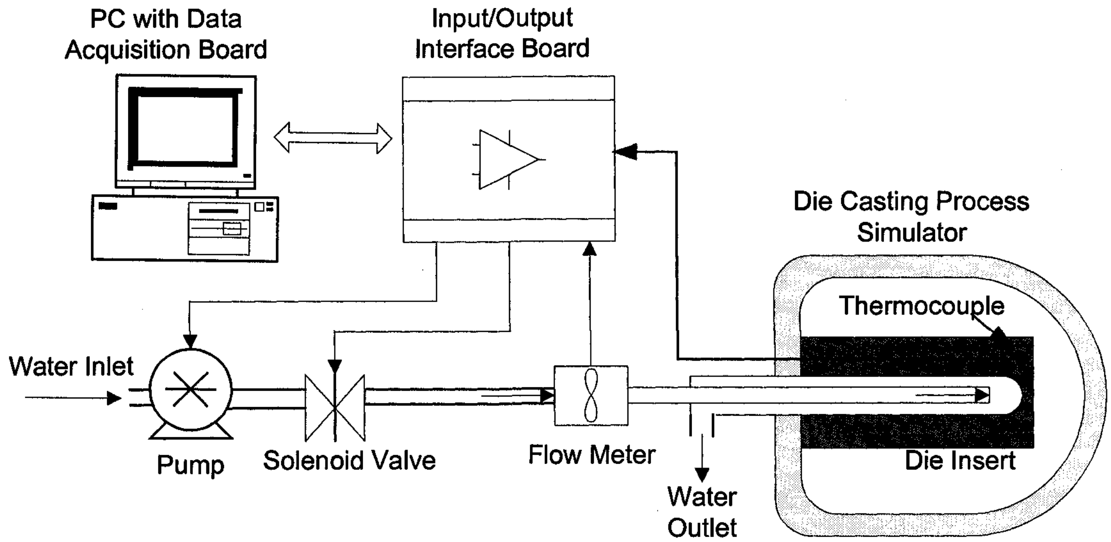


Figure 3.1. Schematic Diagram of the Thermal Management System.

3.2 Die Casting Process Simulator (DCPS)

The schematic diagram of the die casting process simulator is shown in Figure 3.2. Instead of heating the die by directly pouring molten metal onto it, a 3kW furnace is employed to preheat the die insert. A piece of the insert is mounted to a movable supporting rack and pushed into the furnace until it is completely enclosed by the furnace. An insulating plate, made of fiberglass, is placed between the contact surface of the insert and the metal rack. The die insert is preheated in the furnace to a certain temperature, then cooling water is applied to chill the insert. Therefore, one complete experimental cycle includes furnace preheating and insert cooling. The insert cooling stage is used to simulate the solidification stage of a real die casting process. To determine the local die insert temperatures around waterlines, thermocouples are placed in holes drilled from the rear of the insert toward its surface.

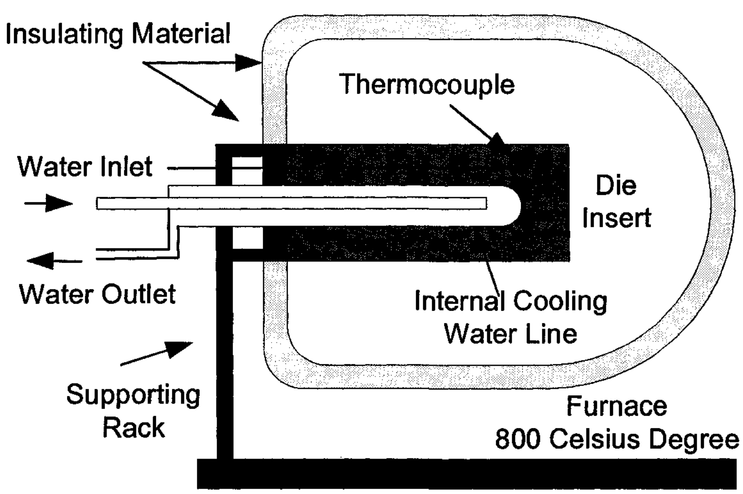


Figure 3.2. Die Casting Process Simulator.

3.3 Selection and Design of Hardware

In the present study, the intelligent real-time monitoring and control system (IRMCS) is required to measure 2 types of analogue signals and output 4 kinds of digital signals. The schematic diagram of the hardware structure is illustrated in Figure 3.3. The inputs to be measured are: temperature signals from thermocouples, flow rate signals from flow sensors. The main function of digital outputs is to control pump speed and solenoid valves, to indicate fault channel and fault code on display panel, and to activate warning light and warning alarm when specified abnormal conditions are detected in the system such as overheating. The desired measurement range for the temperature signal is from $0\,^{o}C$ to $600\,^{o}C$ with resolution $1\,^{o}C$, which covers the window of die temperature variation [2]. For the water flow rate signal, the desired measurement range is from 2.08 L/min (0.55 GPM) to 9.46 L/min (2.5 GPM) with a resolution of 0.0379 L/min (0.01 GPM). The main parts contained in the IRMCS are listed thereinafter, which include the personal computer (PC) with data acquisition board, temperature sensors and signal conditioner, water flow rate sensors, pump and solenoid valves, fault indicators, warning light and alarm, and input/output interface box.

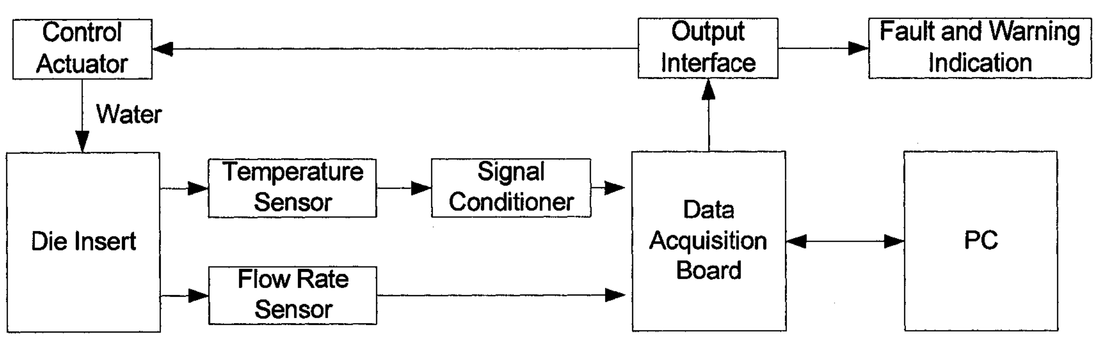


Figure 3.3. Schematic Diagram of the Hardware Structure.

3.3.1 PC Hosted Data Acquisition Board

There are two main functions of a PC hosted data acquisition board used in this system: to convert analogue signals such as temperature and flow rate to their digital counterparts that can be read by PC, and to transmit control commands to actuators such as pump and valves. The Cyberresearch® 12-bit board with 8-channel differential inputs or 16-channel single-ended inputs, part number PCI-DAS 1602/12, is selected due to its functionality and relatively low cost [22]. The smallest voltage that can be detected by this board with an input range from 0 to 10 V can be calculated as:

$$\Delta U = \frac{10V}{2^{12}} = 2.44mV \tag{3.1}$$

3.3.2 Temperature Sensor

Thermocouples and infrared sensors are the most commonly used transducers in die casting processes. Infrared sensors are non-contacting devices that infer temperature by measuring the thermal radiation emitted by a material. Due to the nature of die closure during cavity filling and casting holding, they cannot be used to monitor the die insert temperature continuously in real time, while this is not a problem for thermocouples. They are very rugged and can detect temperatures over a wide range inexpensively. Therefore OMEGA® low noise K-type thermocouple probes, model number HUKMQSS, are employed. It has ground strap connection that provides protection against electrical noise, and its standard error limits are $1.1\,^{o}C$ or 0.4% above $0\,^{o}C$ [23].

3.3.3 Temperature Signal Conditioning

The maximum output voltage of an ANSI standard thermocouple is no more than 60 mV which is too low to be read directly by the data acquisition board, hence AD595D monolithic chips from Analog Device Inc. are utilized as signal conditioners and cold junction compensators for thermocouples. They are instrumentation amplifiers and precalibrated to produce a temperature proportional output of 10 mv/ ^{o}C with calibration error 1 ^{o}C [24]. Differential input is used to reject common-mode noise voltage on the thermocouple leads. The output voltage of this chip is 6161 mV at temperature 600 ^{o}C, so the smallest temperature ΔT that can be detected is

$$\Delta T = \frac{600\ ^{o}C}{6161\ mV} \times 2.44\,mV = 0.24\,^{o}C \tag{3.2}$$

3.3.4 Water Flow Rate Sensor

The cooling water flow rates in die insert are measured by the RotoFlow® flow sensors in Figure 3.4 manufactured by GEMS Sensors Inc [25]. The output range of the sensor is from 0 to 10 VDC proportional to the flow rate from 2.08 L/min (0.55GPM) to 18.9 L/min (5 GPM) as shown in Figure 3.5. This signal can be read directly by the data acquisition board without signal amplifier. The minimum detectable flow rate ΔF can be calculated as

$$\Delta F = \frac{18.9L/\min}{10\times10^{3}mV} \times 2.44mV = 0.0046L/\min \tag{3.3}$$

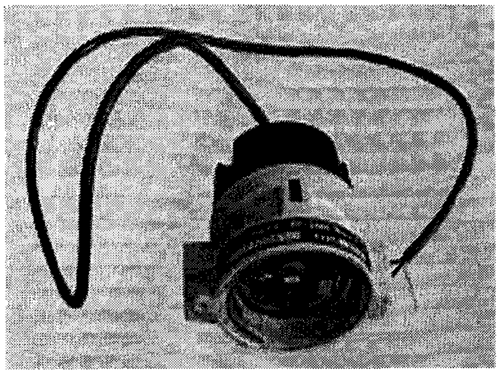


Figure 3.4. Flow Sensor.

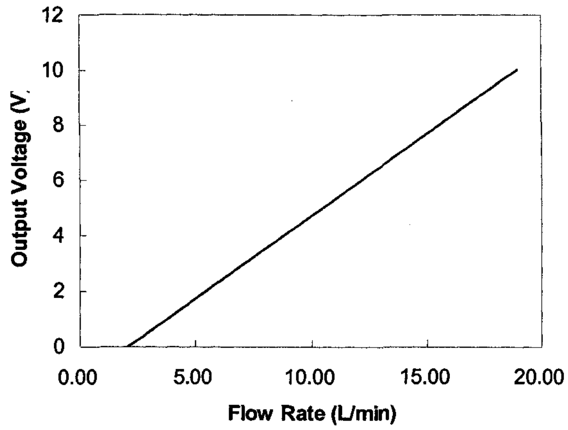
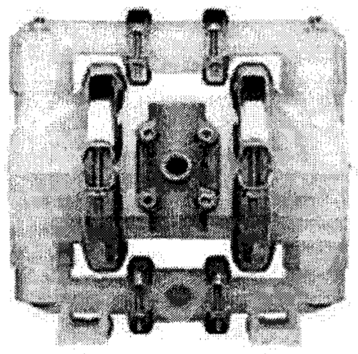
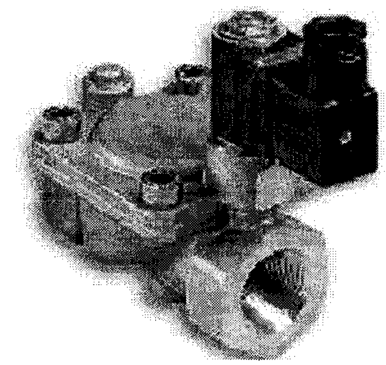


Figure 3.5. Input and Output Signals of Flow Sensor.



(a) Pump



(b) Solenoid Valve

Figure 3.6. Control Actuators.

3.3.5 Pump and Solenoid Valve

An electrical-pulse-controlled, air-operated, double-diaphragm pump (Figure 3.6) made by Wilden® is chosen as control actuator [26]. Through altering the pulse frequency, the pump speed, which is denoted as stroke per minute (SPM), is changed accordingly to vary water flow rates. The pump is capable of increasing water flow rate up to 9.46 liter/minute (L/min) or 2.5 gallon/minute (GPM). Solenoid valves, model number SV-301 from OMEGA, are employed to open or shut down extra cooling waterlines [27].

3.3.6 Fault indicators, warning light and alarm

Two 7-segment light-emitting diodes (LED) are employed to indicate fault channel and fault code on a physical display panel as shown in Figure 3.7 when specified abnormal conditions are detected by the system, such as overheating, low flow rates, and etc. Warning light would be illuminated, and alarm would be raised at the same time.

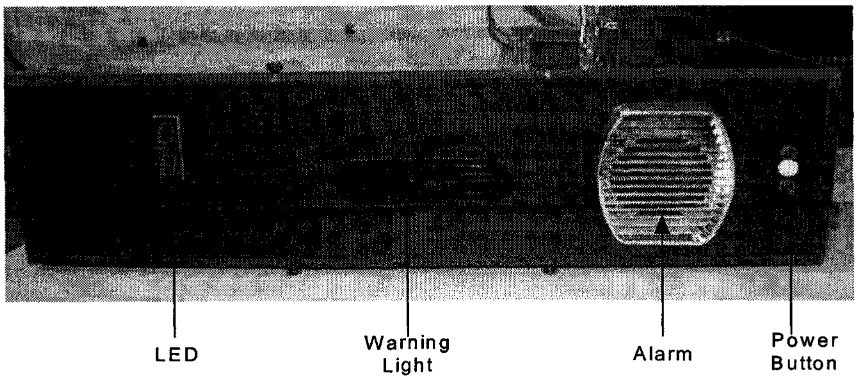


Figure 3.7. Display Panel.

3.3.7 **Input/Output Interface Box**

All the sensor signals and control signals are connected to or from the data acquisition board through the interface box (Figure 3.8). There are four main parts inside the box: (1) adaptor board converts the 100-pin port of the data acquisition board through a ribbon cable to a 100-connector mount for easy wiring; (2) connector board contains several wiring terminals that all sensor and actuator wires go through them; (3) chips and other electric parts are installed on the electric board; and (4) power supply provides +5VDC, +12VDC power sources that the system needs. Digital output pins on the data acquisition board are used as control outputs. Solid-state relays are engaged to isolate the data acquisition board from the electrical circuits of the pump, solenoid valves, warning light and alarm. The typical operating current of solid-state relay in this study is about 20 mA, despite that the maximum digital output current of the data acquisition board is only 2.5 mA, hence the inverter chips (7406) are occupied to provide sufficient current sink to drive the solid-state relay. Decoder chips (7447) are also necessary to drive 7-segment LEDs. The schematic diagram of the electric circuits in the interface box is shown in Figure 3.9.

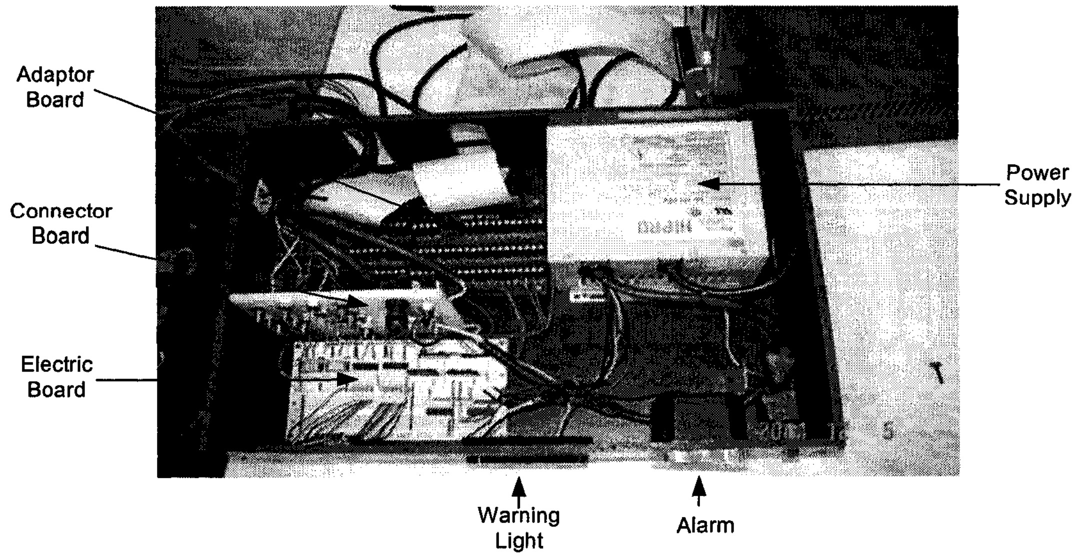


Figure 3.8. Input/Output Interface Box

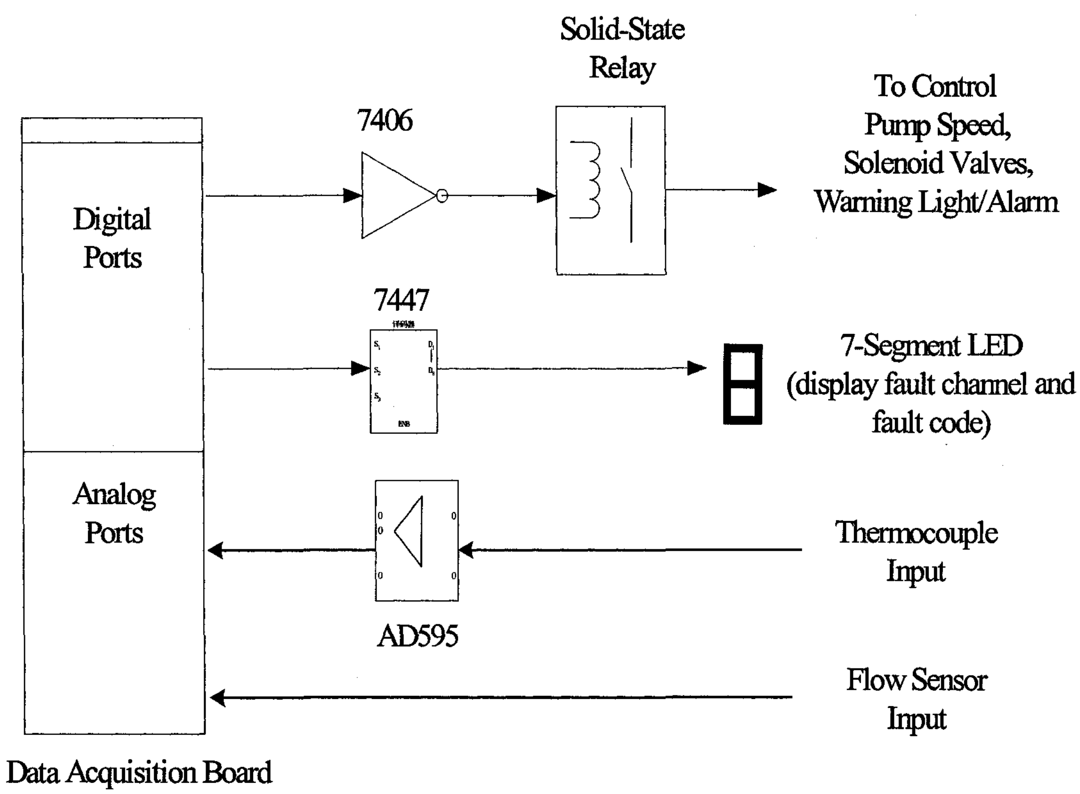


Figure 3.9. Schematic Diagram of Electric Circuits.

3.4 Software Design

The software program (written in C) running on a PC is designed to acquire, monitor, and process data in real-time, and then to send control command according to the temperature fluctuation of the die insert, all through interfacing to the plug-in data acquisition board housed in the PC. The main graphic user interface is divided into five function areas as shown in Figure 3.10. The real-time curves for temperatures and flow rates are displayed in the middle. A bulletin panel on the upper right indicates the status of the system, including temperatures of die insert, positions of solenoid valves, flow rates in cooling waterlines and pump speed. Warning messages can be shown and alarm can be manually turned off on the lower right side of the window. Control functions for pumps and solenoid valves are built on the upper left side where user can select automatic or manual control for each actuator. Some important parameters can be set on the lower left, and other parameters, such as temperature display range, sampling interval, and the size of an average digital filter, can be selected from another pop-up window.

The main program routine consists of five sub-function modules (Figure 3.11): data sampling, data display, data saving, control, and warning modules. The data-sampling module is responsible for acquiring temperature and flow rate data at a sampling rate that can be specified by users. The data-displaying module displays data on the screen in the format of curves and values. The data-saving module saves data to the disk in an ASCII text file format for further analysis. The control module can actuate solenoid valves and adjust pump speed manually or automatically based on its built-in control algorithms.

The warning module can detect the predefined abnormal conditions such as overheating, display fault code on the display panel, illuminate warning light, and sound alarm.

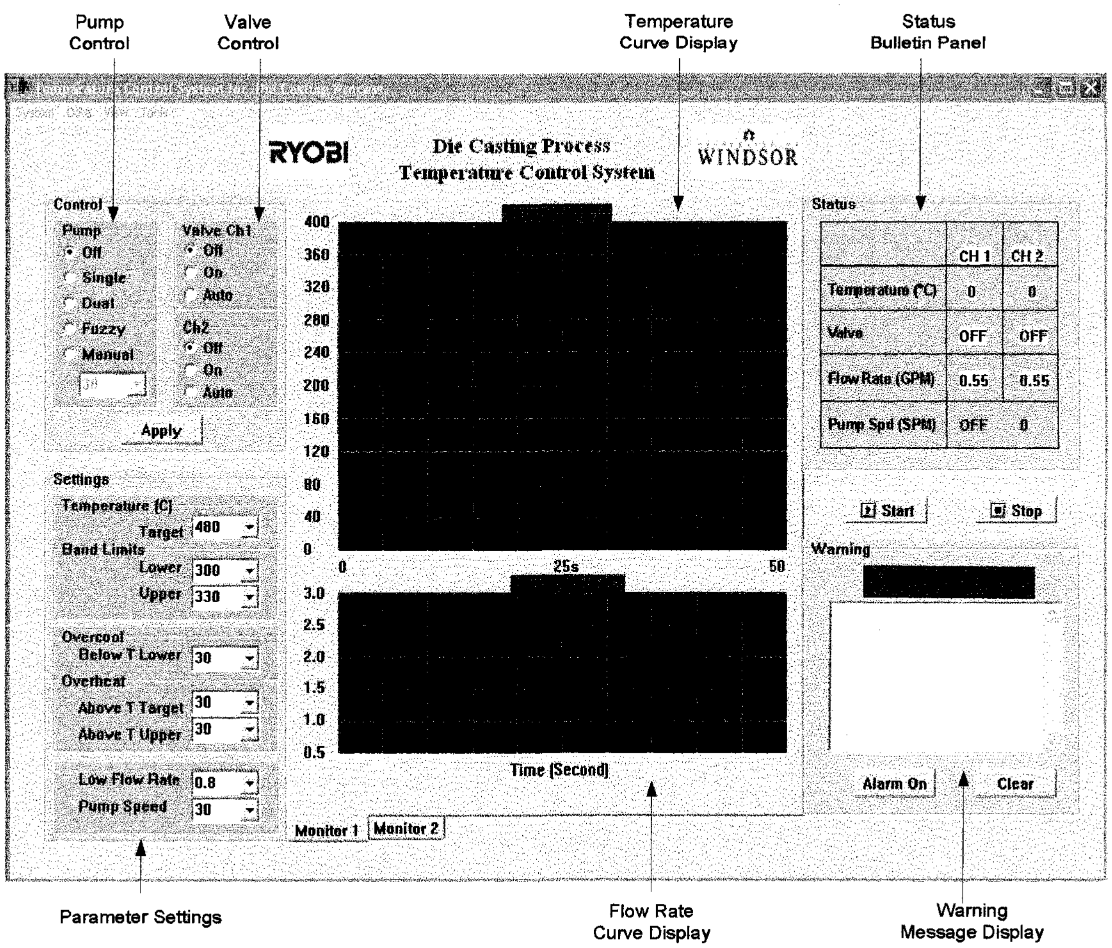


Figure 3.10. Main Graphic User Interface.

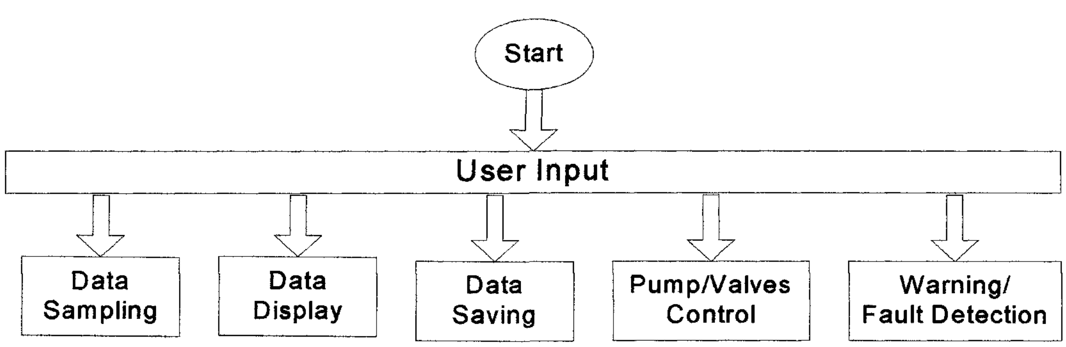


Figure 3.11. Software Subroutines.

Table 3.1. Function Lists for Graphic User Interface.

Name		**Function**	**Control Mode**
Pump Control	Off	Turn off pump manually.	Manual
	Single	Local Temperature controller for single cooling channel.	Automatic
	Dual	Local Temperature controller for dual cooling channels.	Automatic
	Fuzzy	Fuzzy controller to reduce the temperature of hot spots.	Automatic
	Manual	Adjust pump speed manually.	Manual
Valve Control	Off	Turn off solenoid valve manually.	Manual
	On	Turn on solenoid valve manually.	Manual
	Auto	Turn on/off solenoid valve automatically slaved to pump control.	Automatic
Parameter Settings	Target	The maximum temperature that the die insert can be heated up in the furnace before automatic control functions.	
	Lower	Lower bound of temperature band for local controller.	
	Upper	Upper bound of temperature band for local controller.	
	Overcool	Warning if real-time temperature below Lower bound minus digits in this box.	
	Overheat 1	Warning if real-time temperature above Target plus digits in this box.	
	Overheat 2	Warning if real-time temperature above Upper bound plus digits in this box.	
	Low Flow Rate	Warning if real-time flow rate below 0.55 GPM plus digits in this box.	
Others	Start	Start system.	
	Stop	Stop system.	
	Alarm	Press to turn on or turn off the warning function.	
	Clear	Clear warning message window.	

Chapter 4

Thermal Analysis for Single Cooling Channel

In this chapter, correlation between the die insert temperature and cooling water outlet temperature is illustrated under different flow rate. A regression equation has been established, which can be used to deduce local die surface temperatures without destructively inserting thermal sensors into a die.

4.1 Experimental Design

In order to develop a control strategy for die thermal management, it is inevitable to understand the influence of cooling water flow rates on die temperature distribution. Due to the nature of die closure during cavity filling and casting holding, it is very difficult to monitor local die surface temperatures without destructively inserting thermal sensors into a die from its back. The destructive sensing method upsets the operation of die maintenance, and makes casting production costly. However, the measurement of cooling water temperature at the water outlet is relatively without problem. The present work is therefore motivated to investigate the relationship between cooling water outlet temperature and the local die surface temperature under different water flow rate.

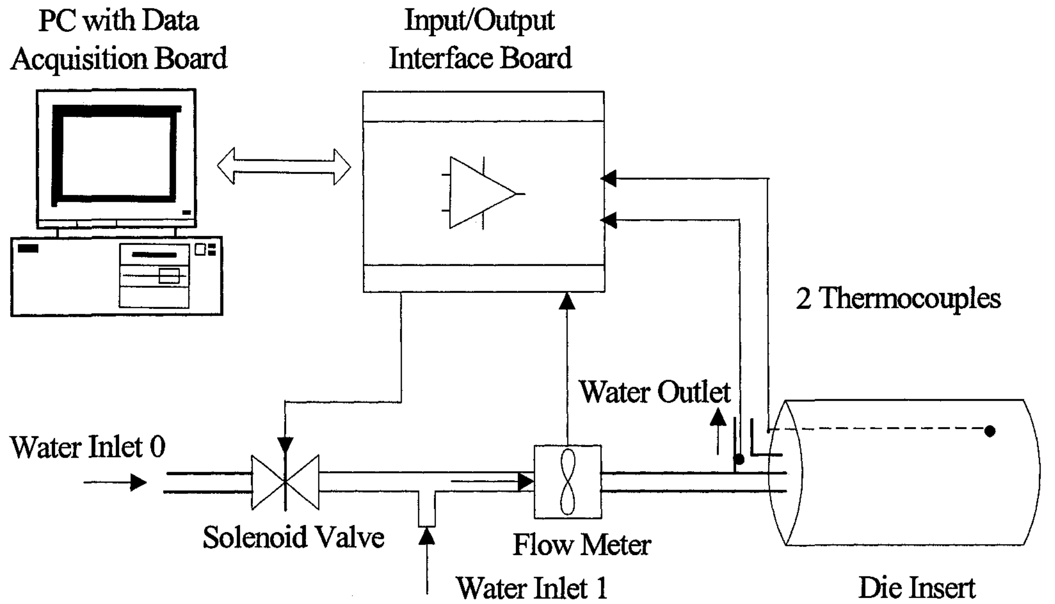


Figure 4.1. Experimental Setup for Single Cooling Channel.

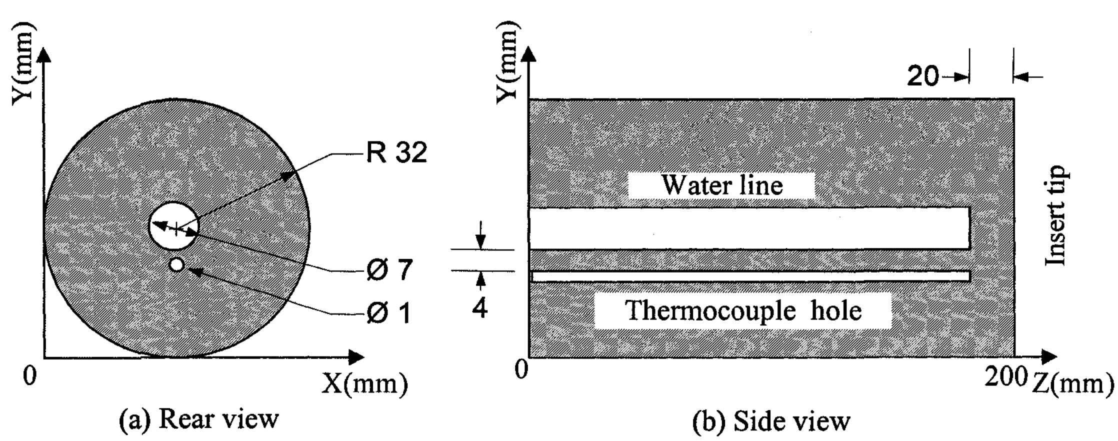


Figure 4.2. Dimension of the Die Insert with Single Cooling Lines.

The experimental setup is shown in Figure 4.1. Two water lines are employed to supply cooling water during the experiments. Water is running with a constant flow rate in Water Inlet 1, while the Inlet 0 is controlled by a solenoid valve to provide extra water flow as required. To determine the die insert temperature and water outlet temperature, one thermocouple is placed in a hole drilled from the rear of the insert toward its surface as illustrated in Figure 4.2. Meanwhile, another thermocouple is positioned into the water line outlet very close to the rear surface of the insert.

4.2 Experimental Procedure

The objective of this experiment is to acquire temperature signals from die insert and cooling water outlet simultaneously. Analysis of the experimental data generates an understanding of how the internal die temperature and the water outlet temperature change with variations in the water flow rate. The experimental procedures are summarized as follows:

Step 1: Keeping a small amount of water (<2.08 l/min (0.55 GPM)) running through water supply line 1.

Step 2: Heating the die insert to 480 ^{o}C, then turn off the furnace.

Step 3: Turning on water supply line 0, increase water flow rate to 3.79 L/min (1.0 GPM), 5.299L/min (1.4 GPM), 6.813L/min (1.8 GPM), and 8.327L/min (2.2 GPM) respectively.

Step 4: Recording experimental data during the cooling stage for 45 seconds.

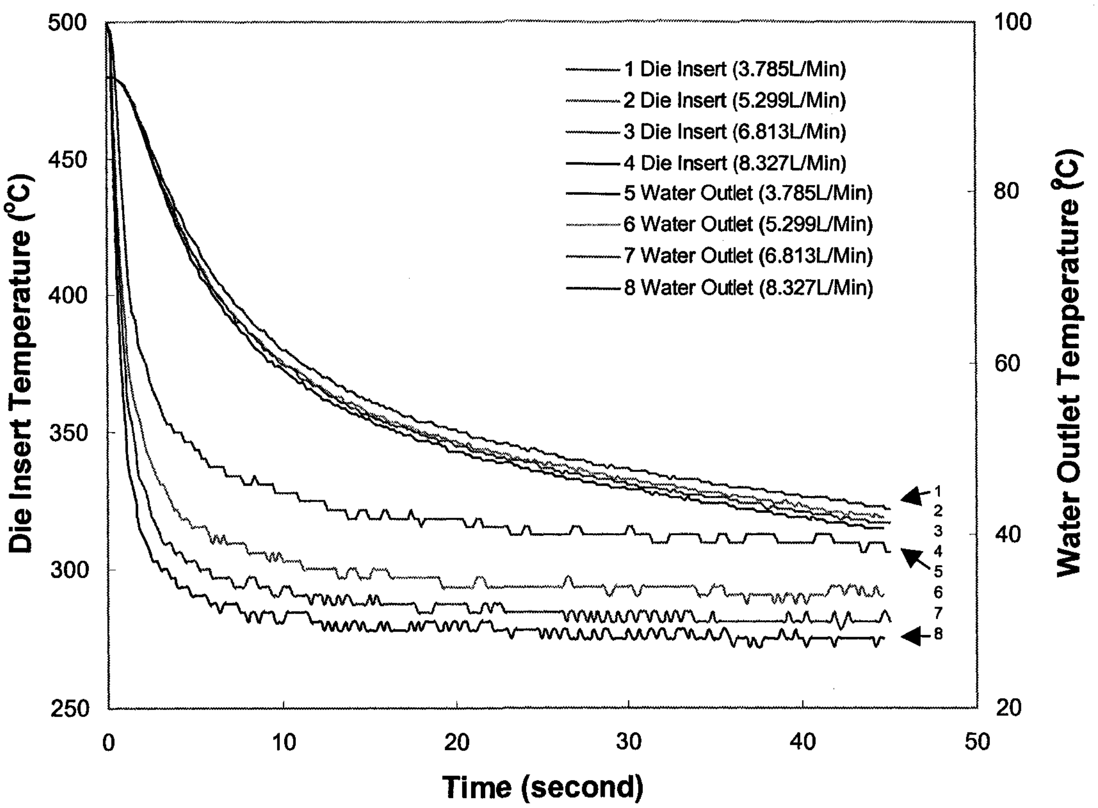


Figure 4.3. Die Insert Temperature and Water Outlet Temperature within 50 seconds.

4.3 Results and Discussions

The experimental results are shown in Figure 4.3. It depicts the variation of the die insert temperature and water outlet temperature with cooling time under different water flow rates for a sampling period of 45 seconds recorded by the IRMCS with a sampling rate of 5 data per second. The primary Y-axis represents the die insert temperature, while the secondary Y-axis stands for the water outlet temperature.

The experimental data show that both the die insert and water outlet temperatures decrease rapidly once extra cooling water is applied. The decreasing rate slows down along with increasing cooling time. As cooling time extends, both the temperatures tend to approach a steady state. An increase in the water flow rate considerably reduces the water outlet temperature at the beginning when extra cooling water is applied. The time for water outlet temperature to reach a steady state is reduced as the water flow rate increases. However, the influence of the water flow rate on the die insert temperature is not as significant as on the water outlet temperature. This is because, in the laboratory setup, the furnace temperature is set at $800\,^0C$ to heat the die insert. During the die cooling stage, even though furnace is turned off, the temperature of the furnace is considerably higher than that of the die inert. As a result, heat continuously transfers from the furnace to the die insert due to the insulated structure of the DCPS, which keeps the die insert temperature relatively high.

In an effort to establish the temperature relation between the die insert and the water outlet, the data presented in Figure 4.3 are re-plotted in Figure 4.4. Examination of Figure 4.4 manifests that an increase in the water flow rates reduces both the die insert temperature and the cooling water outlet temperature simultaneously. The mapping from the water outlet temperatures to the die insert temperatures indicates that the variation of the water outlet temperature with the die insert temperature is primarily one-to-one at a given flow rate. This implies the presence of a correlation between the two temperatures, which are influenced by the applied cooling water flow rates.

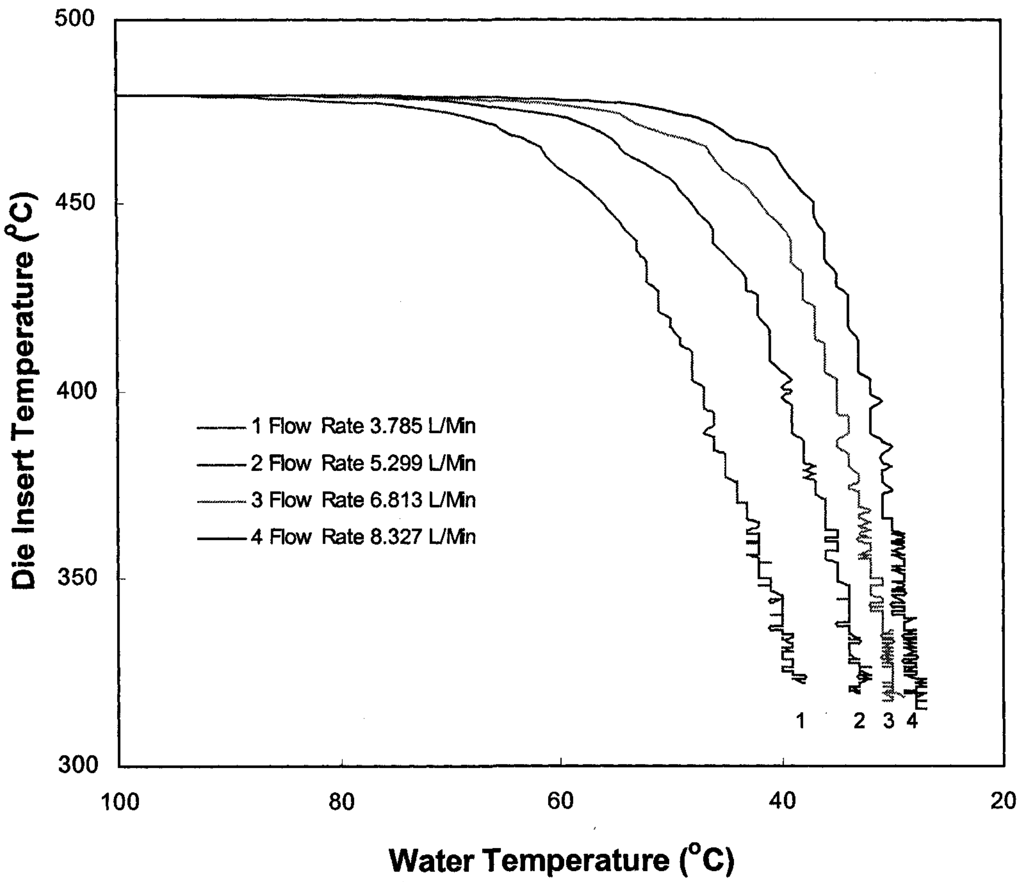


Figure 4.4. Die Insert Temperature and Water Outlet Temperature.

From the given data in Figure 4.4, a regression equation can be deduced by the method of building a linear model as introduced in Section 2.1:

$$T_d = 480 + (630 - 62500\ e^{-\dot{Q}/2})e^{-T_o/10} - (41900 - 3810000\ e^{-\dot{Q}})e^{-T_o/5} \quad (4.1)$$

where T_d and T_o represent the die insert temperature and the water outlet temperature in Celsius degree respectively, $\dot{Q}$ denotes the water flow rate with a unit of liter per minute (L/min). The above equation is valid under the following conditions:

(1) water inlet temperature at 21 ^{o}C;

(2) furnace at 800 ^{o}C, and die insert being heated in the DCPS to 480 ^{o}C before cooling down;

(3) steady flow rate applied from 3.785 l/min (1.0 GPM) to 8.327/min (2.2 GPM);

(4) cooling time or sampling time no more than 45 seconds, and

(5) useful in the following range (between line 1 and line 2 shown in Figure 4.5)

$$-2T_o + 390 \le T_d \le -\frac{5}{7}(T_o - 730) \qquad (4.2)$$

Equation 4.1 correlates the die insert temperature as exponential functions of the water flow rate and water outlet temperature with exponent indices less than zero. The decay of an exponential function is influenced by the value of its exponent. In Figure 4.5, the regression curves, shown as a solid line computed from Equation (4.1), is compares with the original sampled data plotted as symbols. Evidently, an increase in the water flow rate shifts the curve downward. This is due to the fact that the value of the exponent in Equation (4.1) is reduced by the increase of the water flow rate, which makes the decay of these curves fast. Table 4.1 lists the coefficients of the correlation for the experimented flow rates. For all the four tested water flow rates, the correlation coefficients are found to be over 0.90, which provides certain confidence for potential applications. It can also be seen that an increase in the water flow rate reduces the coefficients of the correlation. This observation indicates that the time for the water outlet temperature reaching a steady state shortens as the water flow rate rises.

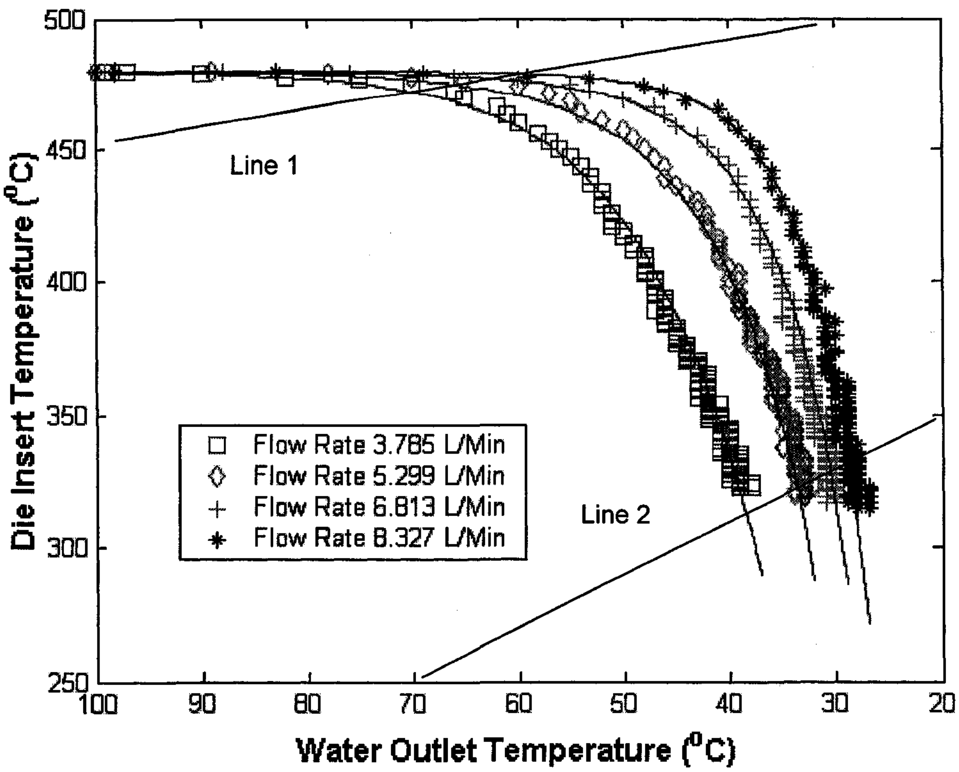


Figure 4.5 Comparisons between Regressive Analysis and Experimental Measurements

Table 4.1. Coefficients of Correlation

Flow Rate (L/Min)	3.785	5.299	6.813	8.327
Coefficient of Correlation	0.9744	0.9202	0.9422	0.9063

A three-dimensional (3-D) plot is displayed in Figure 4.6 for the purpose of demonstrating the combining effect of the water flow rate and the water outlet temperature on the die insert temperature, which is predicted by Equation (4.1). It is evident that a reduction in the water flow rate and the water outlet temperature leads to an inward curl of the 3-D surface. The geometry change of the surface indicates that increasing flow rate lowers both the die insert temperature and the water outlet temperature.

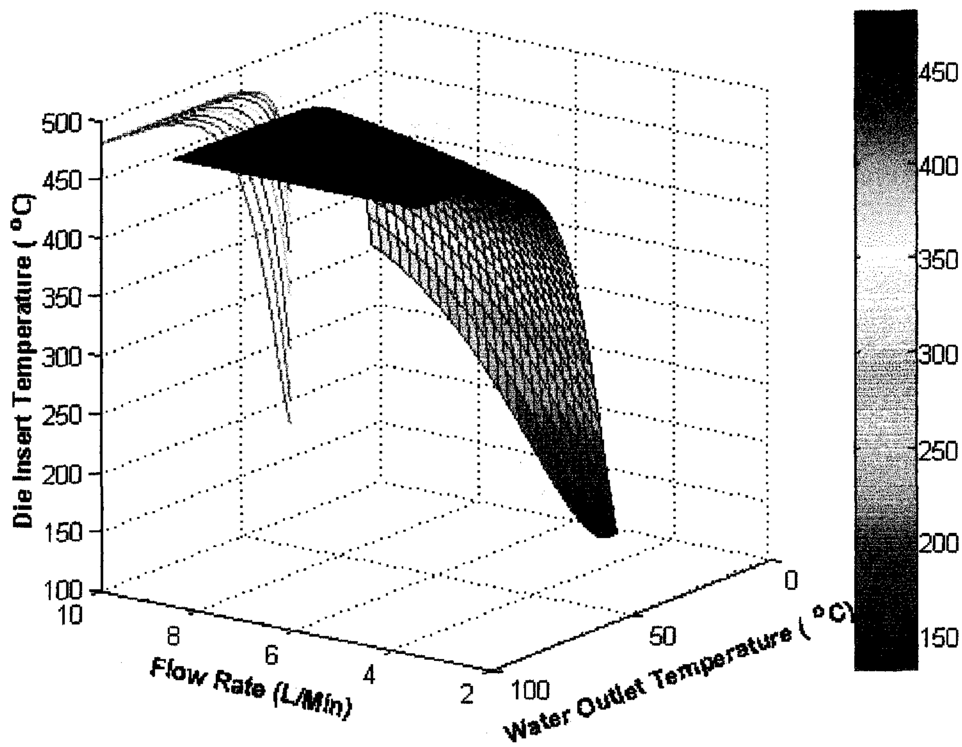


Figure 4.6. Three-Dimensional Plot of Regression Curves.

Chapter 5

Thermal Management for Multiple Cooling Channels

In this chapter new thermal management schemes are proposed for die casting processes with multiple cooling channels. Experiments have been designed and conducted to explore the capacity of the proposed thermal management system in terms of control and monitoring, including effect of an extra cooling line on both flow rate and die insert temperature, local temperature controllers, and a fuzzy controller. Fault detection and warning modules are also integrated into the implemented system.

5.1 Introduction

The proposed on-line thermal management scheme is based on the intelligent real-time monitoring and control system (IRMCS). Its schematic diagram is shown in Figure 5.1. Suppose that Water Inlets 1 and 2 are established cooling water lines in a real die casting insert. In this scheme, extra cooling lines supplying water from Water Inlet 0 controlled by a pump and solenoid valves are hooked up to each existing cooling channel. The system monitors temperature signals from the die inserts and flow rate signals from the cooling lines. Pump and solenoid valves can be actuated either automatically or manually to bring additional cooling water to the die insert. Since it has been demonstrated that an increase in internal cooling water flow rates more rapidly reduces the temperatures of casting dies [2], the desired thermal pattern of the die insert may be realized through this scheme.

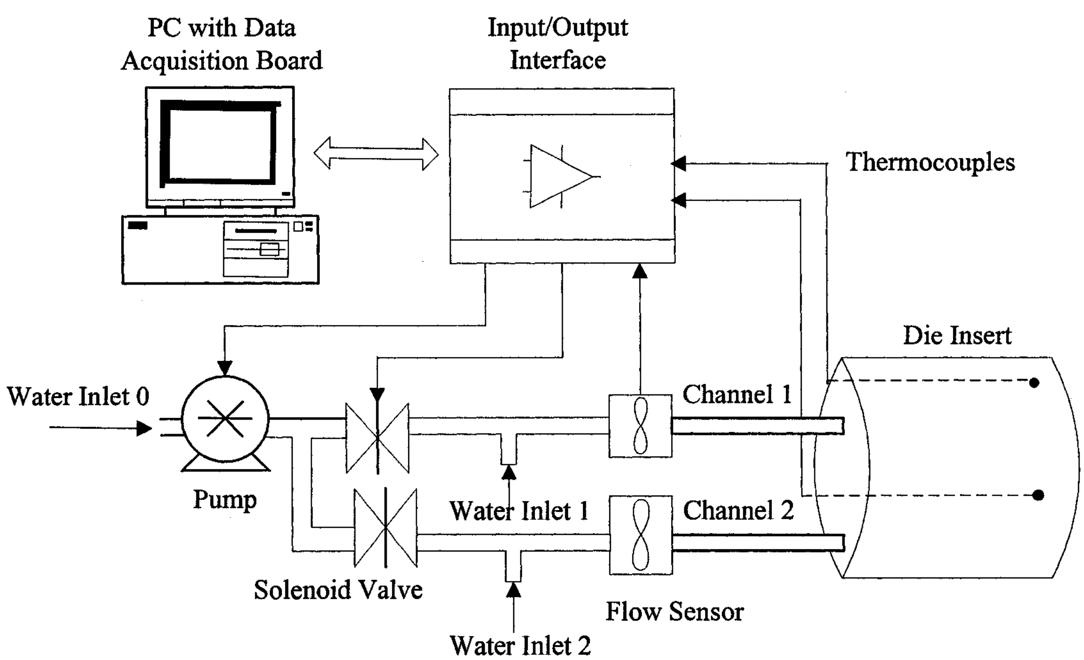


Figure 5.1. Proposed Die Thermal Management System.

In order to examine the performance of the proposed thermal control system, experiment must be designed and conducted to find out what is the better cooling water source for the extra cooling line at Inlet 0; to understand how the total flow rate changes when different pump speeds are being applied; and to comprehend how the temperature of a die varies under various pump speeds. The following experiments are designed and conducted on a die insert with dual cooling channels as shown in Figure 5.2.

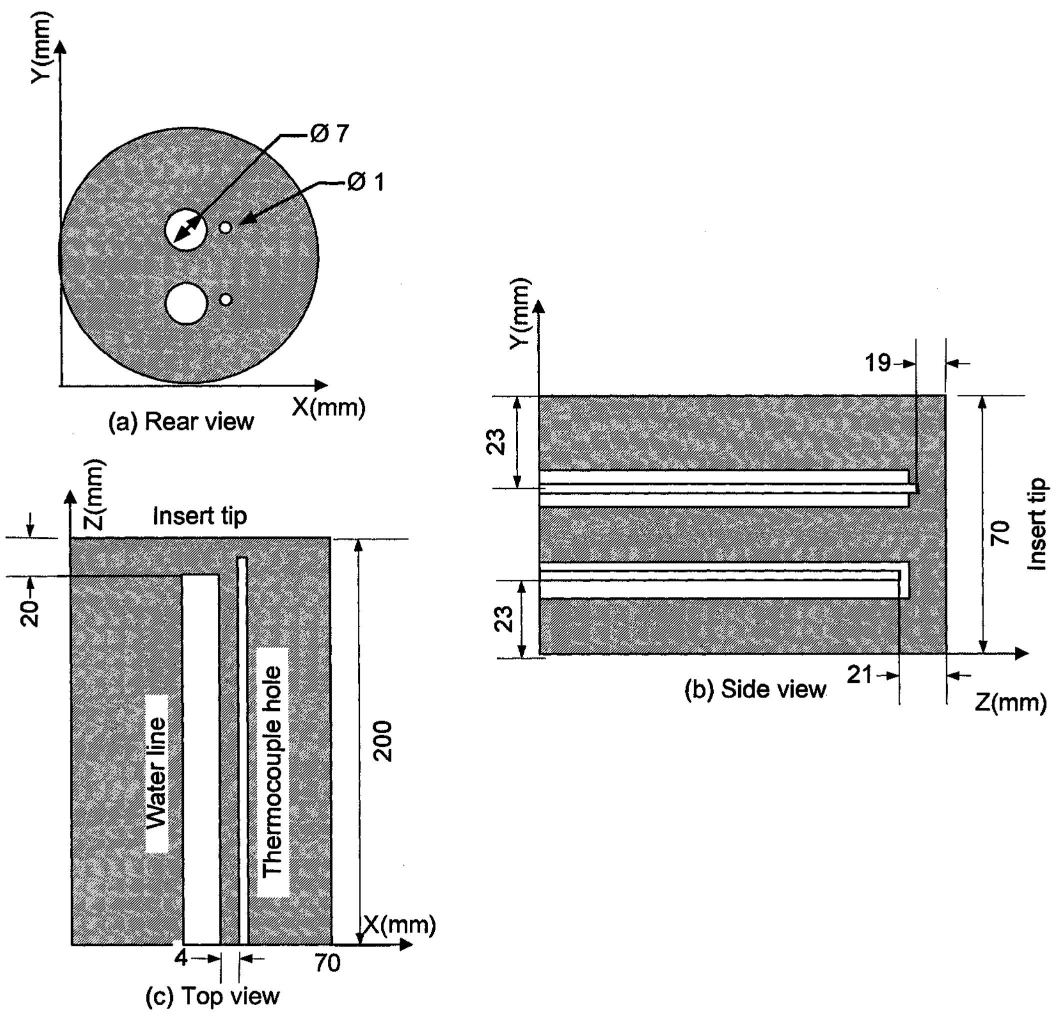


Figure 5.2. Schematic Diagram of a Die Insert with Dual Cooling Lines.

5.2 Experiment A- Effect of Pump Speed on Water Flow Rate

The objective of experiment A is to acquire pump speed signal and flow rate signal simultaneously. Analysis of the experimental data generates an understanding of how the water flow rate changes with variations in the pump stroke rate.

5.2.1 Experimental Procedure

In this experiment, two configurations at Water Inlet 0 are tested for comparison. It is sufficient to conduct the Experiment A only on one channel since these two channels are almost identical.

(1) Configuration 1: Water Inlet 0, and Inlet 1 are all set at a fixed flow rate.

(2) Configuration 2: Water Inlet 1 is set at a fixed flow rate, while a water tank is employed at Inlet 0 as an extra water supplying source.

For both configurations, their experimental procedures can be summarized in Figure 5.3.

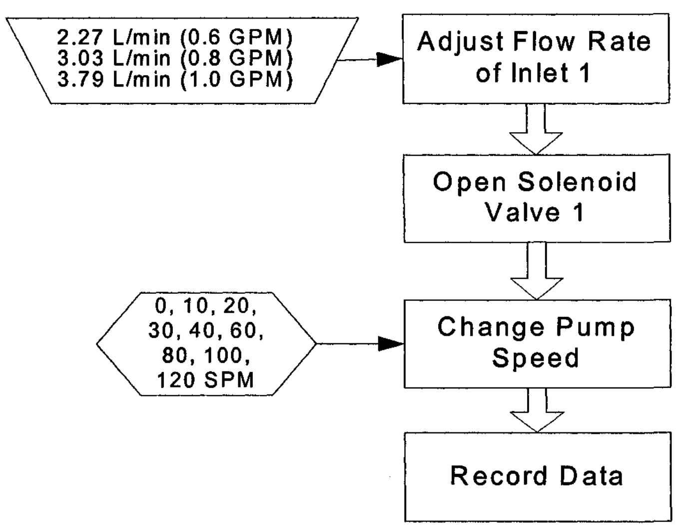


Figure 5.3. Procedure of Experiment A.

5.2.2 Results and Discussions

Since pump stroke rate is controlled by electrical signals that are generated from the digital ports of the data acquisition board, the total flow rates measured by the flow sensor in real-time are saw-toothed curves. Figure 5.4 gives the comparison of flow rate signals between pump speed at 0 SPM and at 40 SPM.

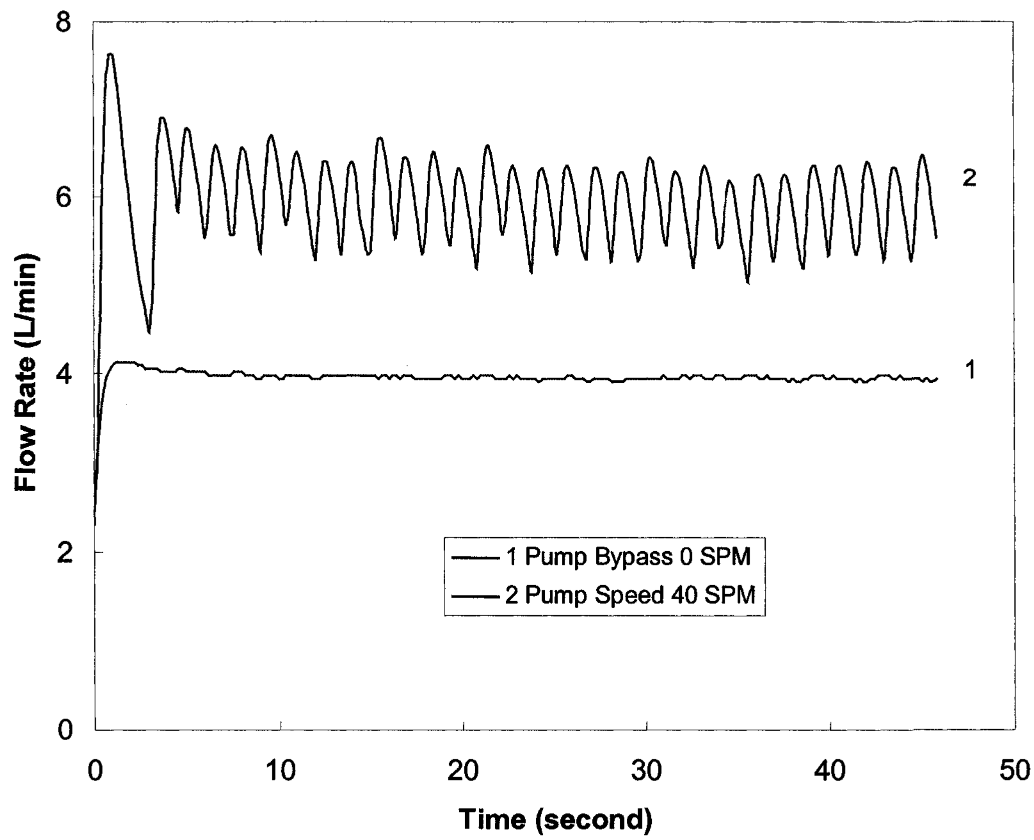


Figure 5.4. Real-time Flow Rate Signal.

Due to the fluctuation of flow rate introduced by pump, the total average flow rate $\dot{Q}_T$ is adopted thereinafter to describe the cooling water flow rate that enters the die insert. Obviously, its function can be written as

$$\dot{Q}_{Ti} = \frac{\sum_{j=0}^{k} \dot{Q}_{Mi}(k)}{k} = f(\dot{Q}_0, \dot{Q}_i, \sigma) \tag{5.1}$$

where $\dot{Q}_{Ti}$ is the total average flow rate at the i-th cooling channel, $\dot{Q}_{Mi}(k)$ is the flow rate of i-th cooling channel at discrete time k measured by flow rate sensor; $\dot{Q}_0$

represents the flow rate at Water Inlet 0, $\dot{Q}_i$ stands for the flow rate at Water Inlet i, and σ denotes other uncertain parameters in the system, such as water line diameter. The flow rate change introduced by the extra cooling water line can be defined as

$$\Delta\dot{Q}_{Ti} = \dot{Q}_{Ti} - \dot{Q}_i \tag{5.2}$$

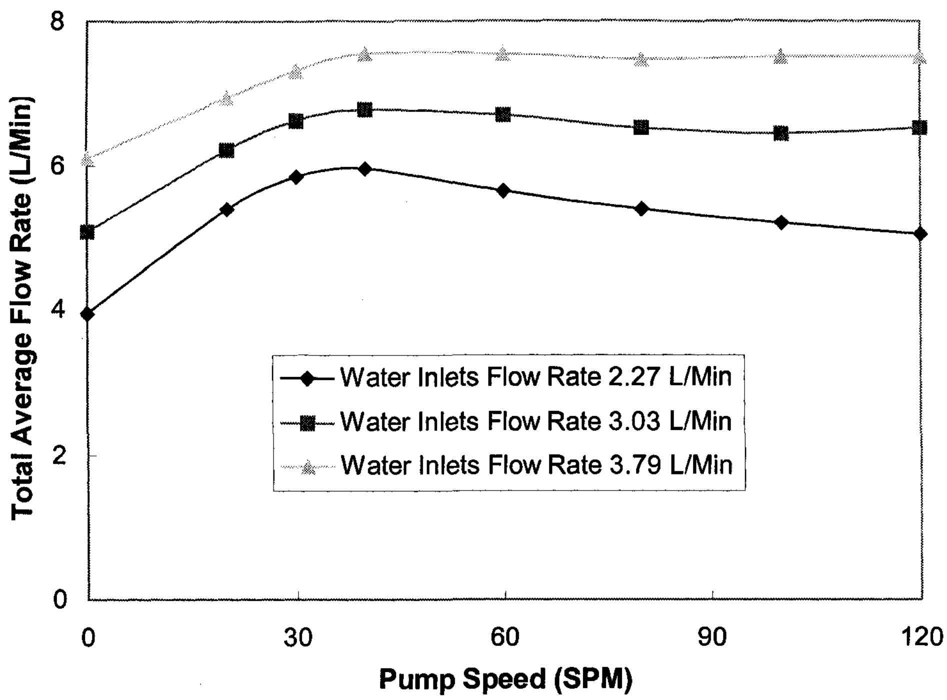


Figure 5.5. Total Average Flow Rate with Configuration 1.

Table 5.1. Flow Rate Change with Configuration 1

Pump Speed (SPM)	0	20	30	40	60	80	100	120
Inlets 0 and 1 at 2.27(L/min)	1.67	3.11	3.57	3.68	3.38	3.11	2.92	2.77
Inlets 0 and 1 at 3.03(L/min)	2.05	3.19	3.60	3.75	3.68	3.49	3.41	3.49
Inlets 0 and 1 at 3.79(L/min)	2.31	3.15	3.52	3.75	3.75	3.68	3.71	3.71

The effect of pump speed on average flow rate with configuration 1 is depicted in Figure 5.5. The water flow rates at Inlet 0 and Inlet 1 are set at 2.27 L/min (0.6 GPM), 3.03 L/min (0.8 GPM), and 3.79 L/min (1.0 GPM) respectively, and pump speed increases from 0 SPM to 120 SPM for each flow rate. The experimental results show that once the pump speed is raised from 0 SPM, the total average water flow rate increases. But it soon reaches a saturation point at pump speed 40 SPM. The total average water flow rate tends to remain almost constant even as the pump speed further increases beyond the saturation point. This early saturation phenomenon may be caused by the limited water pressure at Inlet 0, the pump capacity, and the size of water line. Table 5.1 shows the flow rate change calculated from equation 5.2. Although this configuration is able to boost total average flow rate, the early saturation may restrict its application for control purpose due to the fact that the margin for pump to function is quite small.

The influence of pump speed on the total average flow rate with configuration 2 is illustrated in Table 5.2 and Figure 5.6, where a water tank is employed at Inlet 0 to feed the pump, and flow rate at Inlet 1 is fixed at 2.27 L/min (0.6 GPM). It is evident that the total average flow rate increases as the pump advances to a higher speed, and the saturation point is shifted to 100 SPM. This configuration is capable of increasing the amount of cooling water considerably, and can provide more margins for pump control than that of Configuration 1. Hence it is more desirable in applications.

Table 5.2. Effect of Pump Speed on Total Average Flow Rate with Configuration 2

Pump Speed (SPM)	0	10	20	30	40	60	80	100	110	120
Flow Rate (L/min)	2.29	4.18	5.55	6.29	6.85	7.72	8.84	9.21	9.40	9.42

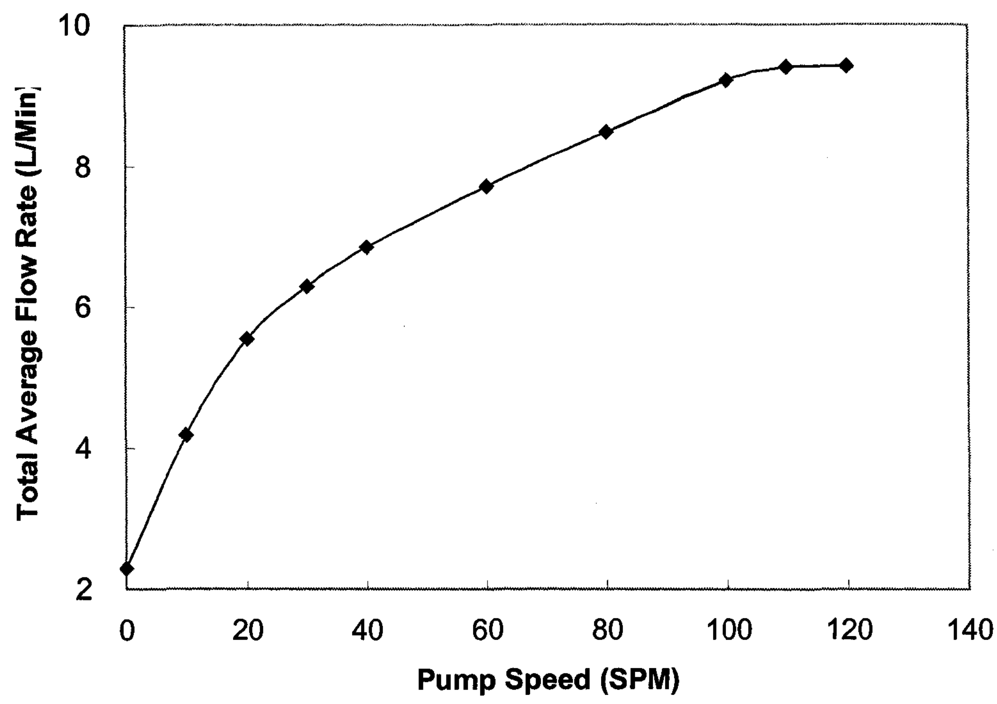


Figure 5.6. Total Averaged Flow Rate with Configuration 2.

5.3 Experiment B- Effect of Pump Speed on Die Insert Temperature

This experiment is designed to acquire necessary data for investigating the relationship between pump speed and die insert temperature under the above mentioned two configurations.

5.3.1 Experimental Procedure

Experiments are conducted following the procedures described in Figure 5.7. It should be mentioned that the pump is started earlier in order to accumulate certain pressure in cooling line before die insert reaches $480°C$, at which point the inlet 1 and solenoid valve 1 are opened to introduce cooling water into the die insert.

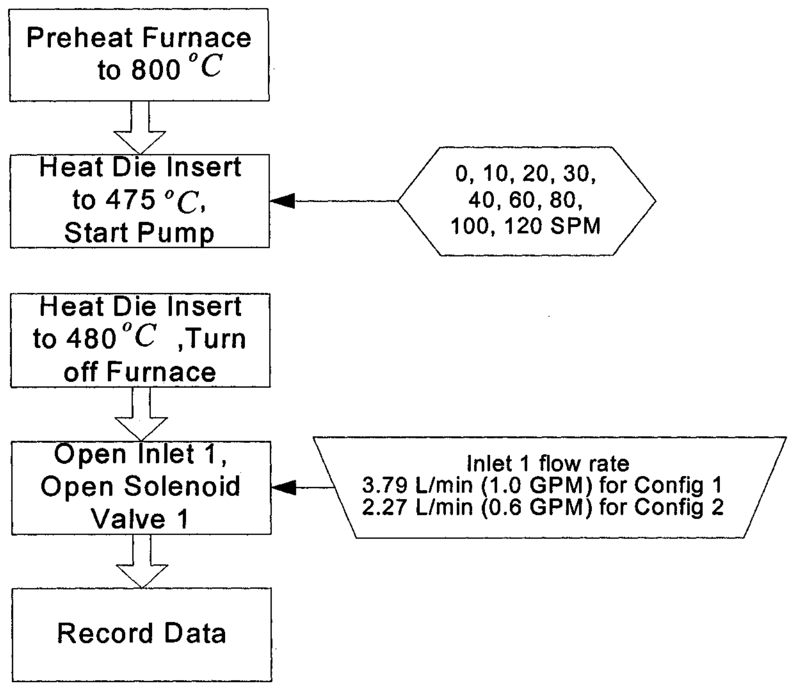


Figure 5.7. Procedure of Experiment B.

5.3.2 Results and Discussions

The experimental results are plotted in Figure 5.8 for die insert temperature with Configuration 1. The primary Y-axis represents die insert temperature, while the secondary Y-axis stands for cooling water flow rate that enters the die insert. Flow rate of Inlets 0 and 1 is fixed at 3.79 L/min (1.0 GPM), and pump speed is adjusted at saturation point, 40 SPM, to provide the maximum amount of cooling water. Die insert temperature curve without the extra cooling line is also plotted for the purpose of comparison. It is apparent that die insert temperature drops more rapidly when there is extra cooling water running through the die insert. The temperature difference between these two curves is about 12 $^{\circ}C$ at the 45th second after the onset of cooling. This difference of 12 $^{\circ}C$ is the observed maximum temperature drop that Configuration 1 can achieve due to the saturation.

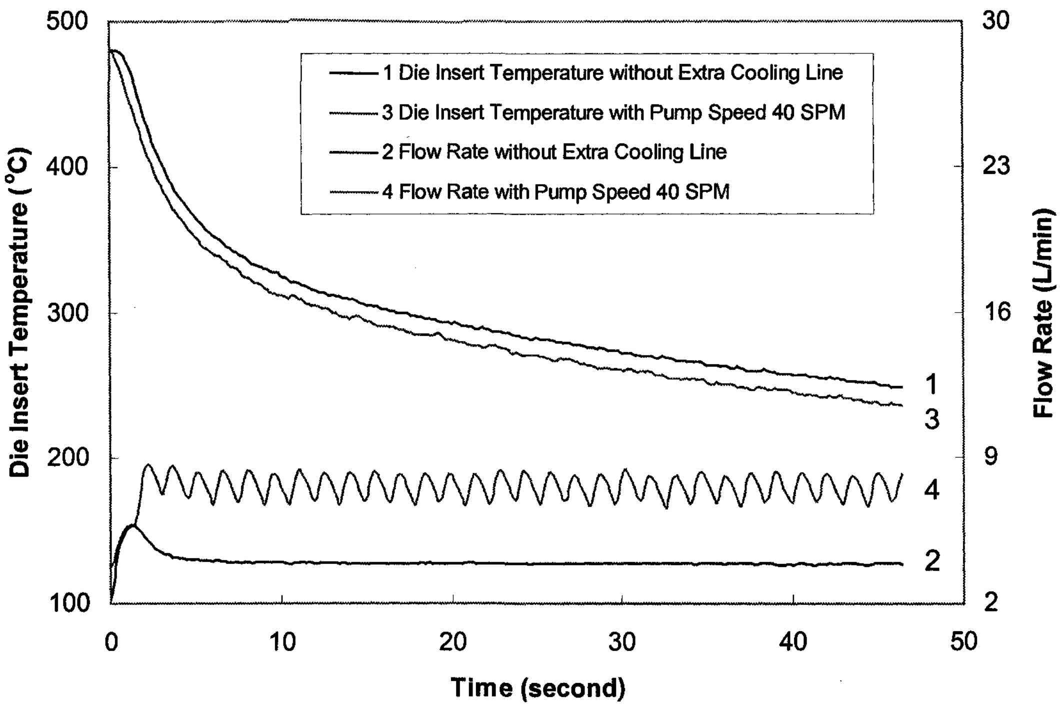


Figure 5.8. Die insert temperature with configuration 1.

The effect of pump speed on die insert temperature with configuration 2 is depicted in Figure 5.9. It is obvious that an increase in the pump speed, which results in an increase in the total average flow rate as verified by Experiment A, can considerably reduce the die insert temperature. Table 5.3 lists the die insert temperatures after cooled for 45 seconds under various flow rates. It indicates that the maximum temperature drop among these curves is about $37\,^{o}C$, significantly larger than that of Configuration 1, which means that employing an extra water supplying source to feed the pump enhances the capability of system for die thermal management. Table 5.3 is also plotted in Figure 5.10 where a linear equation can be used to fit the scattered data point:

$$T_d(45) = -0.384 * V + 268 \qquad (5.3)$$

where $T_d(45)$ represents the die insert temperature in Celsius degree at the 45^{th} second after the cooling stage starts, and V denotes the pump speed in SPM. This equation shows that control of die insert temperature can be accomplished by proper control of pump speed during cooling stage. Consequently the desired thermal pattern of the die insert becomes achievable.

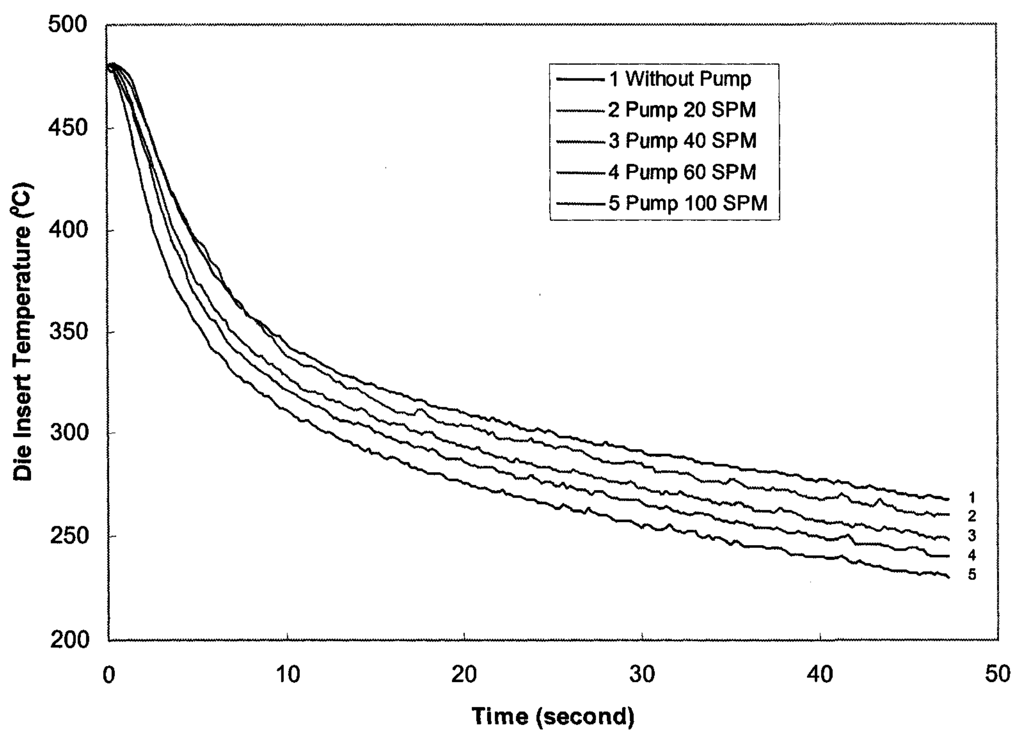


Figure 5.9. Die Insert Temperature with Configuration 2.

Table 5.3. Die Insert Temperature after Cooled for 45 seconds.

Pump Speed (SPM)	0	10	20	30	40	60	80	100
Die Temperature (°C)	269	265	261	256	251	244	236	232

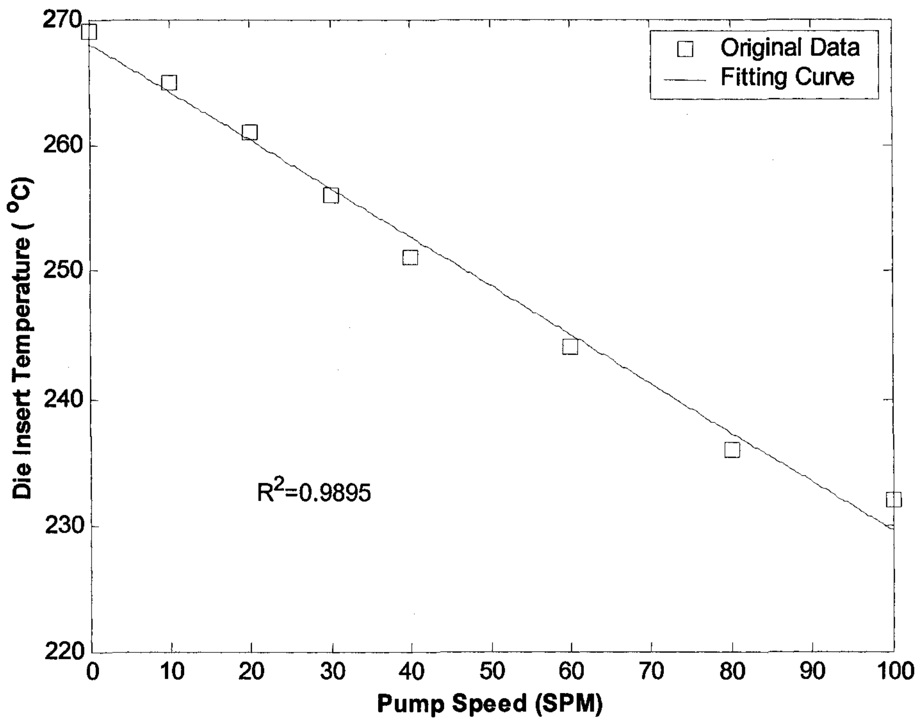


Figure 5.10. Die Insert Temperature after Cooled for 45 seconds.

5.4 Experiment C- Local Temperature Controller

In this Section, the design of a local controller is discussed based on the law of die temperature fluctuation. It is worth pointing out that the temperature range, instead of individual temperature point is used to adjust the actuating signals in the control algorithm. Therefore the implementation of this controller enables the system to limit local die insert temperatures in a given band.

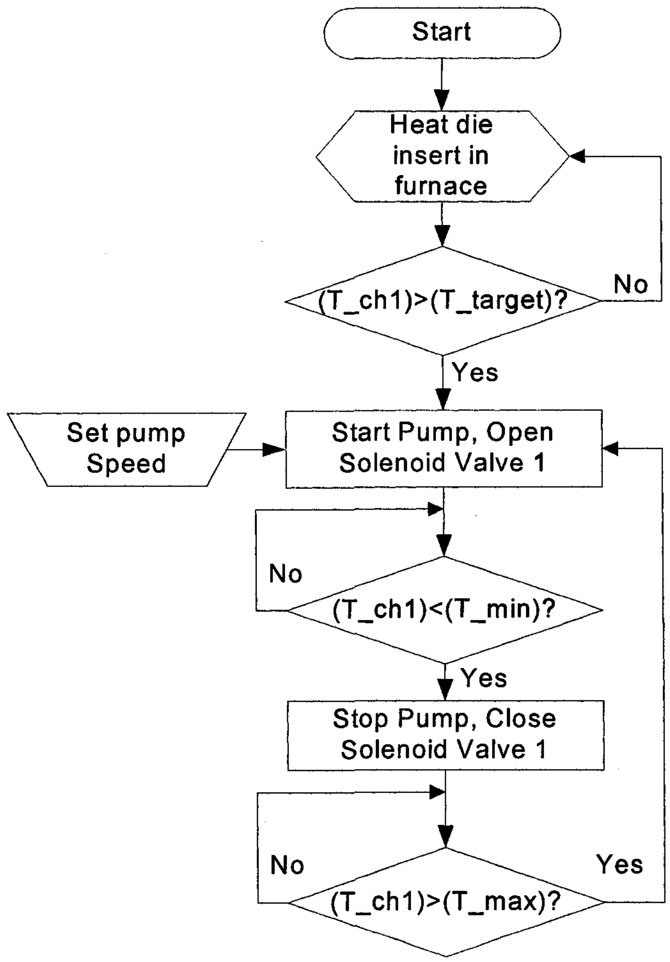


Figure 5.11. Flow Chart of Control Algorithm for Single Cooling Channel.

5.4.1 Control Algorithm for Single Cooling Channel

The flow chart of the control algorithm for single cooling channel is illustrated in Figure 5.11 where T-ch1 is the real-time die insert temperature measured in channel 1, T-target denotes the maximum temperature that the die insert is heated up in the furnace, T-max is the upper bound of the desired temperature range, and T-min stands for the lower bound, respectively. All these parameters can be adjusted through graphic user interface (GUI) as mentioned in Chapter 3. It is obvious that the following inequality exists for the three critical temperature points

$$(T\_target) \geq (T\_\max) > (T\_\min) \tag{5.4}$$

The die insert is first heated in the furnace. The control algorithm monitors its temperature, and starts pump and opens solenoid valve 1 automatically to cool down the die insert when its temperature reaches T-target. The cooling stage stops and resumes automatically at T-min and T-max, respectively. Through this control strategy, the die temperature can be limited to a specific range $[T\_\min, T\_\max]$.

Table 5.4. Parameter Settings for Single Cooling Channel.

T-target	T-max	T-min	Pump Speed	Flow rate	
				Inlet 0	Inlet 1
$480^{\circ}C$	$330^{\circ}C$	$300^{\circ}C$	30 SPM	3.79 L/min (1.0 GPM)	0

5.4.2 Experimental Results and Discussions

The parameter settings for this experiment are listed in Table 5.4. The experimental results are shown in Figure 5.12, where the solid lines represent the die insert temperature and flow rate respectively. It is evident that the cooling stage starts immediately after die insert temperature is heated up to $480^{\circ}C$. When the die insert is chilled by cooling water to about $300^{\circ}C$, the pump stops and solenoid valve 1 closes. A new cooling cycle resumes after the die insert temperature reaches about $330^{\circ}C$.

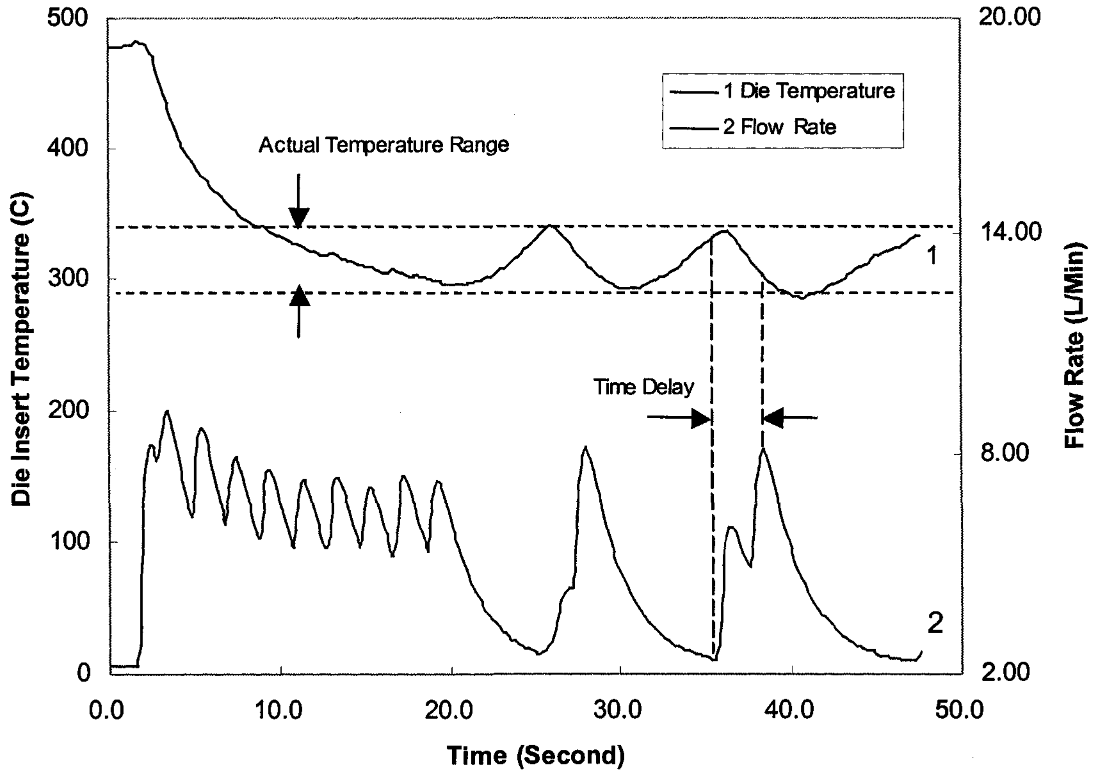


Figure 5.12. Local Temperature Controller for Single Cooling Channel.

It should be mentioned that time delay is observed in this experiment when flow rate is climbing up to its peak value boosted by the pump. As shown in Figure 5.12, the two vertical lines, which represent the time interval for flow rate to reach its crest, intercept the temperature curve of the die insert at 330 ^0C and 303 ^0C respectively. However, due to the time delay, the actual temperature range that the die insert can be controlled falls into [288^0C, 338^0C] as shown between the two horizontal lines. Overshooting happens for both upper temperature bound and lower bound. The time delays may be caused by the following reasons:

1) Pump is a typical inertial component that needs time from energizing the coil to reach its full operation state;
2) Since Windows XP used in this control system is a general purpose operating system that does not have real-time performance, the latency is unpredictable when a number of other tasks are being performed either on the front window or under the table, if not both;
3) Sensors also introduce delays into the system. Flow sensor used in this application, for example, measures the flow rate by counting how many revolutions the rotor vanes swivel in a given time. To measure the flow rate accurately, the friction between the rotor and its support must be as small as possible, i.e., the rotor must rotate very smoothly. However, the rotor can not stop abruptly due to the inertia when the water flow is cut off from the measured waterline. As a matter of fact, delay also occurs when the rotor starts due to the inertia.

Perhaps the most effective approach to reduce the effect of time delay and overshooting on a desired temperature range is to take into consideration the overshooting resulting from the time delay during the input of desired temperature settings. For example, if the desired temperature band is $[T\_\min, T\_\max]$, because it is software selectable in this design, the temperature range can be input through the graphic user interface (GUI) as

$$[T\_\min + \Delta T_1, T\_\max - \Delta T_2] \tag{5.5}$$

where ΔT_1 and ΔT_2 are positive biasing numbers. By adjusting these two parameters, the desired temperature range can be achieved.

Although the experiment is conducted on the condition that water inlet 0 is fixed at a constant flow rate (configuration 1), similar results are expected with configuration 2 where a water tank is employed at inlet 0.

5.4.3 Experiment for Dual Cooling Channels

The control algorithm for the solenoid valves in dual cooling channels remains the same, but it is slightly different for pump control. The pump stops when both solenoid valves are closed, and starts when either valve is opened. The flow chart of the control algorithm is illustrated in Figure 5.13, and the parameter settings are listed in Table 5.5. The experimental results are shown in Figure 5.14, where no biasing parameters are used. By adjusting the water flow rate, the temperatures of the die insert at both channels are controlled in a desired band of approximate 300-330 $^{\circ}C$. It is evident that this controller can handle the tasks for confining the temperatures in two channels within a given boundary.

Table 5.5. Parameter Settings for Dual Cooling Channels.

T-target	T-max	T-min	Pump Speed	Flow rate		
				Inlet 0	Inlet 1	Inlet 2
480 $^{\circ}C$	330 $^{\circ}C$	300 $^{\circ}C$	30 SPM	7.58 L/min (2.0 GPM)	0	0

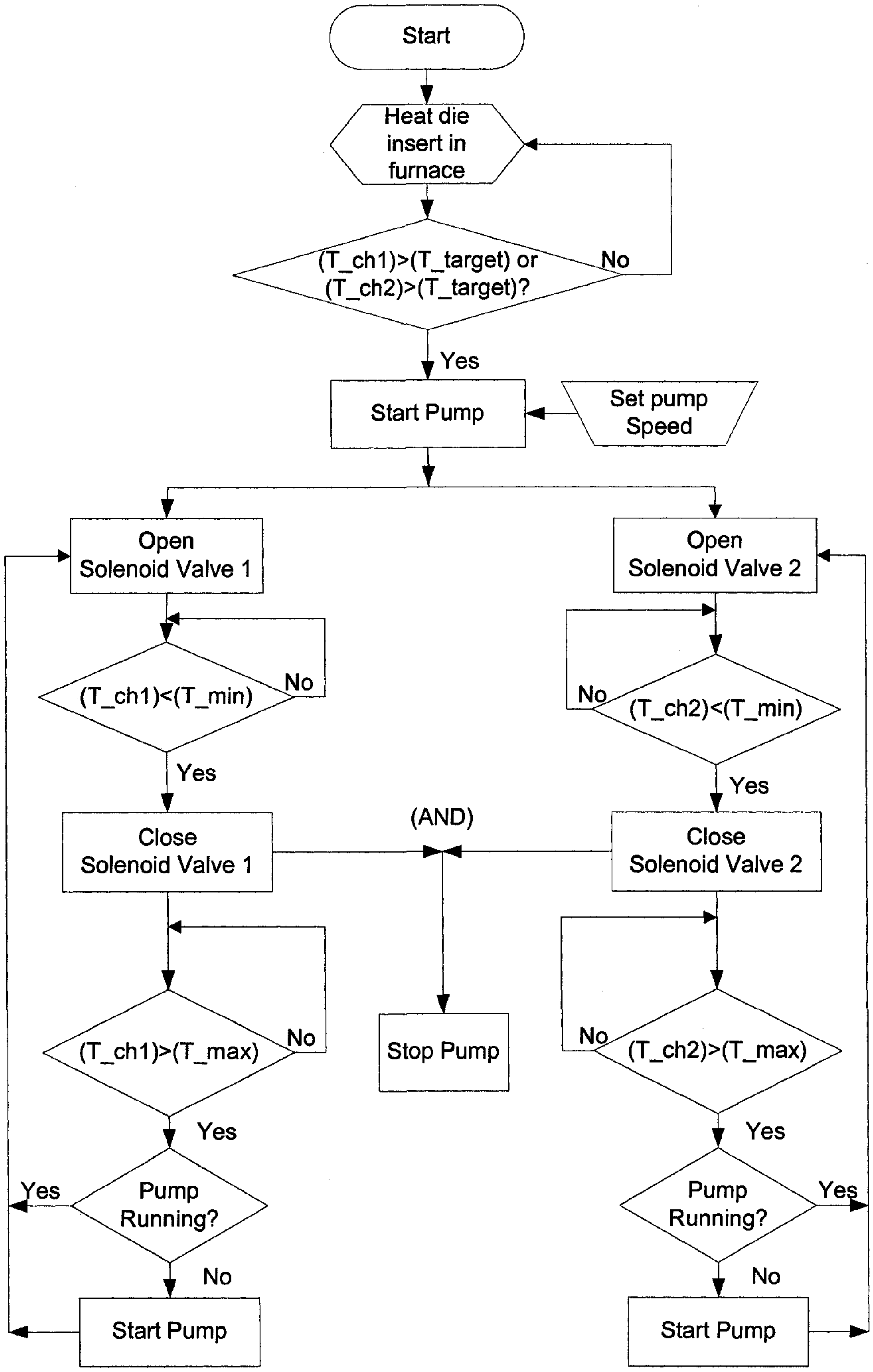


Figure 5.13. Flow Chart of Control Algorithm for Dual Cooling Channels.

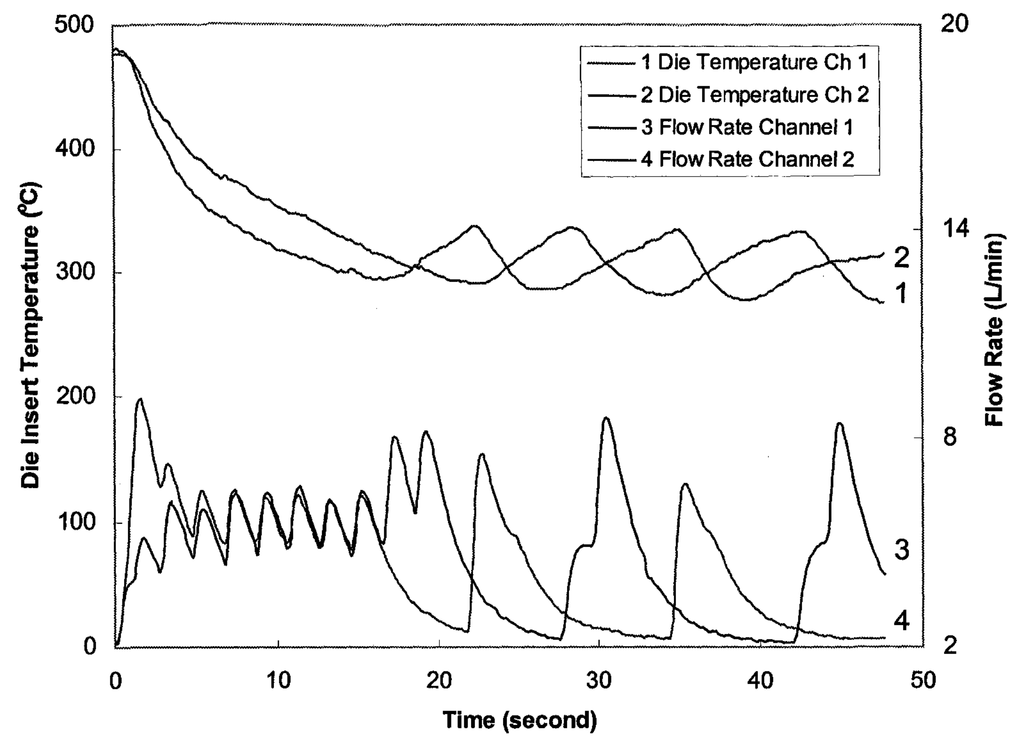


Figure 5.14. Local Temperature Controller for Dual Cooling Channels.

5.5 Experiment D- Fuzzy Controller

Very often a system model is necessary for traditional control approach to design an effective controller. However, die temperature distribution depends on various die design and process variables for which first principles models are very difficult to be obtained. Fuzzy control represents and implements human's knowledge about how to control a system without having a model of the system, therefore it is implemented in this project to reduce the temperature of the hot spots. Of course, the basic knowledge about the system is still needed to achieve precise control, such as the mapping range from input to output. Suppose that in Figure 5.1 one channel with a higher temperature than that of the other is the location of a hot spot during die casting, a fuzzy controller can be designed to effectively reduce the temperature of the hot spot.

5.5.1 Controller Design

In the present study, die insert temperature from both channels is chosen as the controller's feedback signal. Their difference (denoted as *error*) and the error change are input variables to the fuzzy controller, while the output variables are pump speed and on/off status of solenoid valves. The controller is designed based on a sample interval equal to 0.2 second.

Input Variables

At any time $t = k$, the crisp error can be written as

$$e(k) = T_1(k) - T_2(k) \tag{5.6}$$

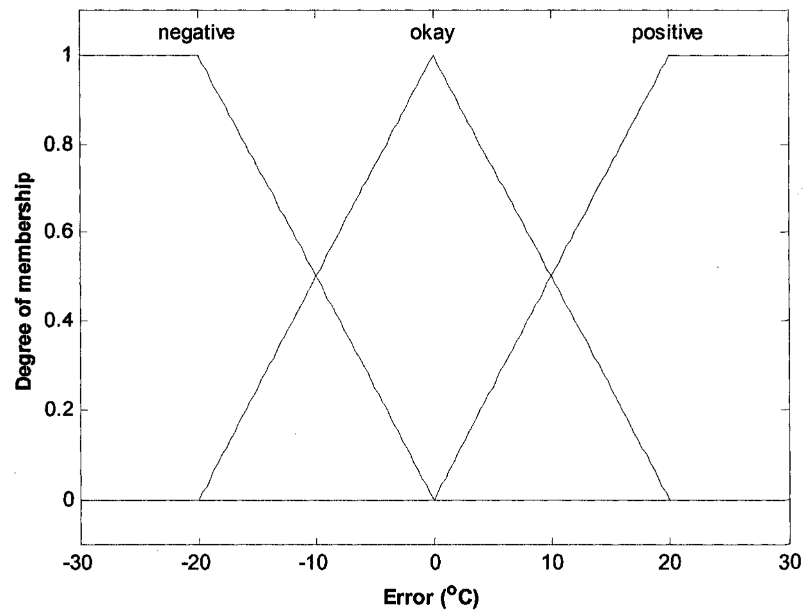
where $T_1(k)$ and $T_2(k)$ represent measured temperatures in Celsius Degree at channel 1 and channel 2 respectively. Its membership function μ_e is defined as

$$\text{Low: } \mu_e = \begin{cases} 1 & \text{if } \; e \le -20 \\ -\frac{1}{20}e & \text{if } -20 < e \le 0 \end{cases}$$

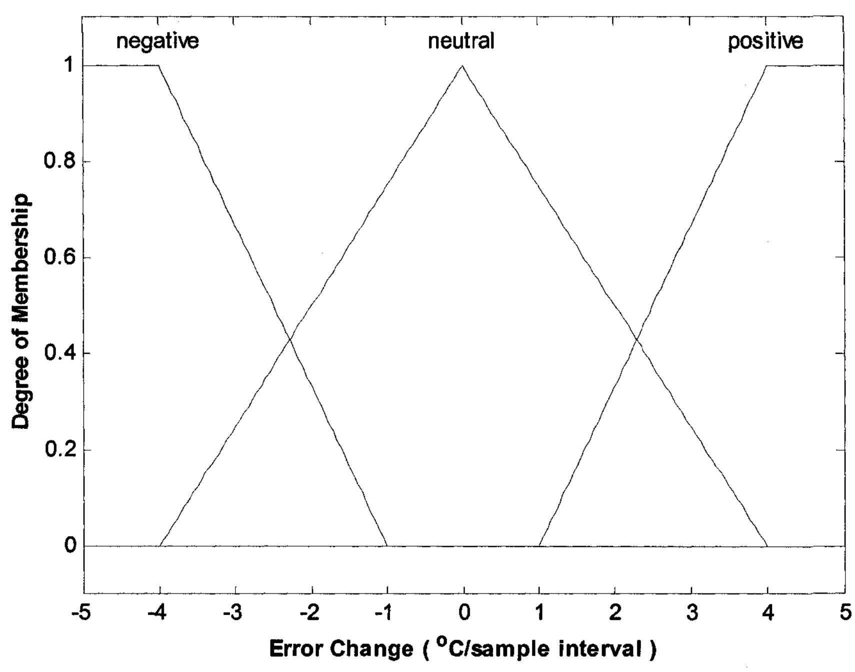
$$\text{Okay: } \mu_e = \begin{cases} \frac{1}{20}(e+20) & \text{if } -20 \le e \le 0 \\ -\frac{1}{20}(e-20) & \text{if } \; 0 < e \le 20 \end{cases} \tag{5.7}$$

$$\text{High: } \mu_e = \begin{cases} \frac{1}{20}e & \text{if } 0 \le e < 20 \\ 1 & \text{if } \; e \ge 20 \end{cases}$$

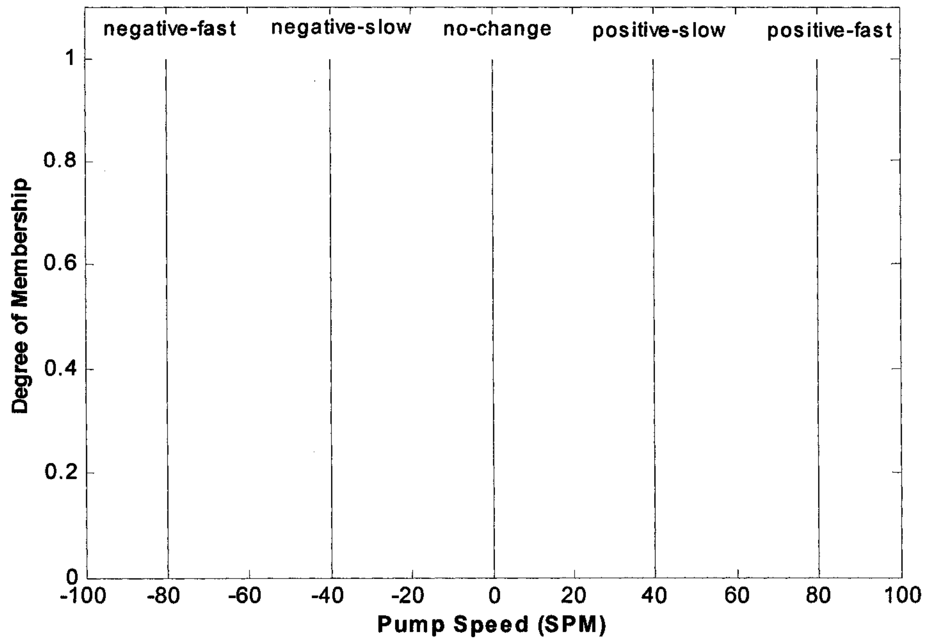
Its membership function is plotted in Figure 5.15 (a).



(a) Error



(b) Error Change



(C) Pump Speed

Figure 5.15. Membership Functions of Input and Output Variables.

At any time $t = k$, the crisp change in error is the difference between present error and the error in the previous time step $t = k - 1$, namely

$$\Delta e(k) = e(k) - e(k-1) \tag{5.8}$$

Its membership function $\mu_{\Delta e}$ can be given as

$$\text{Negative: } \mu_{\Delta e} = \begin{cases} 1 & \textit{if} \quad \Delta e \le -4 \\ -\frac{1}{3}(\Delta e + 1) & \textit{if} -4 < \Delta e \le -1 \end{cases}$$

$$\text{Neutral: } \mu_{\Delta e} = \begin{cases} \frac{1}{4}(\Delta e + 4) & \textit{if} -4 \le \Delta e \le 0 \\ -\frac{1}{4}(\Delta e - 4) & \textit{if} \quad 0 < \Delta e \le 4 \end{cases} \tag{5.9}$$

$$\text{Positive: } \mu_{\Delta e} = \begin{cases} \frac{1}{3}(\Delta e - 1) & if\, 1 \leq \Delta e < 4 \\ 1 & if \quad e \geq 4 \end{cases}$$

This function is also plotted in Figure 5.15 (b).

Output Variables

The system has three output variables, namely, pump speed and on/off action of two solenoid valves. In order to simplify the analysis, the three parameters can be integrated symbolically into one control variable by deliberately introducing positive and negative sign to pump speed. For example, if the pump speed is denoted as "-80" SPM when solenoid valve 1 is opened, solenoid valve 2 is closed, and the real pump speed is at 80 SPM, then "+40" SPM means that solenoid valve 1 is closed, solenoid valve 2 is opened, and the real pump speed is at 40 SPM. It should be mentioned that solenoid valve 1 and 2 could not be opened simultaneously due to the fact that there exists one and only one hot spot in this experiment. The membership function u of the output variables can be written as follows where $\delta(t)$ is $\delta - function$.

$$\begin{gathered} \text{Negative fast: } u = -80\delta(t) \\ \text{Negative slow: } u = -40\delta(t) \\ \text{No change: } u = 0 \\ \text{Positive slow: } u = 40\delta(t) \\ \text{Positive fast: } u = 80\delta(t) \end{gathered} \tag{5.10}$$

This function is also plotted in Figure 5.15 (c).

IF/THEN Rules

Although the controller's knowledge base, the set of if/then rules, is formulated in a human-like language, they have rigorous mathematical foundations involving fuzzy sets and relations. The desirable behavior of the system can be formulated as a collection of rules combined by the connective ELSE listed below

Rule 1: IF error is okay, THEN pump speed is no change, ELSE

Rule 2: IF error is negative, THEN pump speed is positive fast, ELSE

Rule 3: IF error is positive, THEN pump speed is negative fast, ELSE (5.11)

Rule 4: IF error is okay, and error change is negative, THEN pump speed is positive small, ELSE

Rule 5: If error is okay, and error change is positive, then pump speed is negative small.

where Mamdani min for fuzzy implication is used to interpret the connective ELSE as OR operator. A graphic representation of the control surface is given in Figure 5.16.

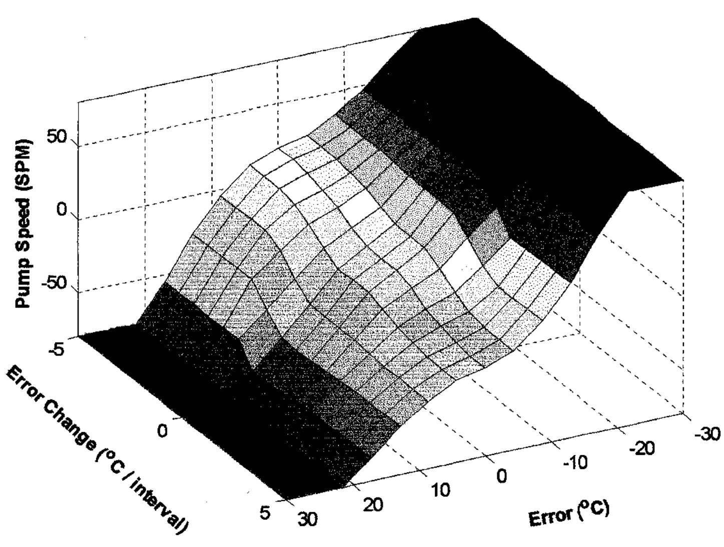


Figure 5.16. Control Surface of the Fuzzy Controller.

Since Windows is not a real-time operating system, the latency is unpredictable when the software transmits a control command to the pump and solenoid valves. This can cause the malfunction of the controller. Therefore it is impractical to change pump speed frequently after each sampling. In computer implementations, fuzzy values can be quantized and stored in memory in the form of a look-up table as shown in Table 5.6, which is also a simplified list. The computer acquires temperature signals from each cooling channel at a fixed sampling rate; then the error and error change are calculated after each sample. For a given temperature error and its derivative, the corresponding pump speed can be found from the look-up Table 5.6.

Table 5.6. Look-up Table for Fuzzy Controller.

Error (^oC)		<-10	[-10, -3]	-2	-1	0	1	2	[3, 10]	>10
Error Change (^oC/interval)	-2	80 SPM	40 SPM	10 SPM	10 SPM	10 SPM	0	0	-40 SPM	-80 SPM
	-1	80 SPM	40 SPM	10 SPM	0	0	0	0	-40 SPM	-80 SPM
	0	80 SPM	40 SPM	10 SPM	0	0	0	-10 SPM	-40 SPM	-80 SPM
	1	80 SPM	40 SPM	0	0	0	0	-10 SPM	-40 SPM	-80 SPM
	2	80 SPM	40 SPM	0	0	-10SPM	-10 SPM	-10 SPM	-40 SPM	-80 SPM

5.5.2 Experimental Results and Discussions

In order to simulate hot spot in this experiment, one channel is chilled first when die insert reaches certain temperature, then fuzzy controller kicks in to adjust the temperature difference between two channels. The experimental procedures are summarized as follows:

Step 1: Heat the die insert to 480 ^oC;

Step 2: Open water inlet 2 to chill channel 2 first for about 3 seconds;

Step 3: Open water inlet 1 and start fuzzy controller simultaneously;

Step 4: Record experimental data during the cooling stage for about 50 seconds.

The experimental result is depicted in Figure 5.17 where the flow rates of water inlet 1 and 2 are fixed at about 2.27 L/min (0.6 GPM), and a water tank is used at water inlet 0 for pump intake. The temperature difference between two channels is about 93 ^oC when the cooling stage for channel 1 starts after the channel 2 has been chilled for about 3 seconds.

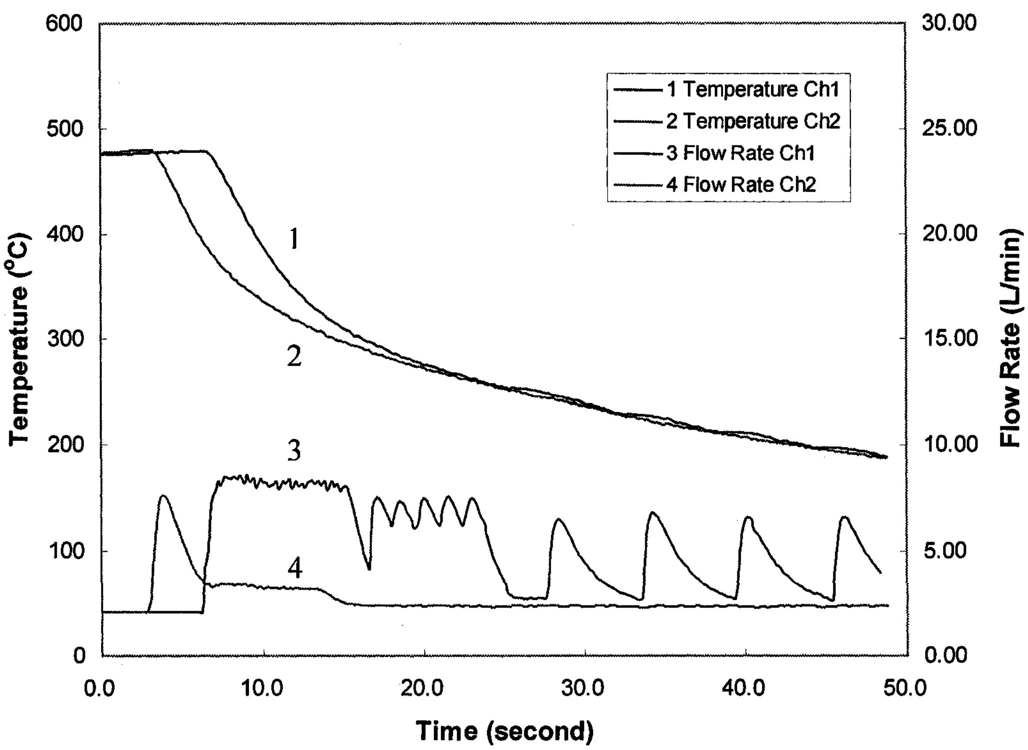


Figure 5.17. Fuzzy Controller.

It can be seen from the figure that pump speed varies to charge additional water into channel 1 according to the temperature difference and error change. At the beginning of the cooling stage for channel 1, the pump is operated at the maximum speed 80 SPM where the temperature difference is positively large. The average flow rate in channel 1 is about 8.80 L/min compared with 2.70 L/min in channel 2. The temperature of channel 1 decreases more rapidly than that of channel 2 due to extra water running through it, hence the pump speed decreases accordingly. It is obvious that the fuzzy controller can effectively adjust the flow rates and consequently can reduce the temperature of the hot spot.

5.6 Fault Detection and Warning

The fault detection and warning module is also integrated into the system to monitor the status of each channel. When an abnormal situation occurs, such as die insert overheating, low flow rate due to the clog of cooling channels, and etc., warning light would be illuminated and alarm would be raised at the same time to notify the operator. As mentioned in Chapter 3, two LEDs are employed to indicate the location of a fault channel, and present fault codes on the display panel located on the front of the interface box. The primary fault codes are divided into five categories:

Code 1. Overheating

$$\text{True if} \begin{cases} T > T\_target + \Delta T_3 & or \\ T > T\_\max + \Delta T_5 \end{cases} \tag{5. 12}$$

Code 2. Overcooling

$$\text{True if } T < T\_\min + \Delta T_4 \tag{5. 13}$$

Code 3. Thermocouple Disconnected

True if measured temperature equals to zero. (5.14)

Code 4. Low Flow Rate

True if pump starts and both solenoid valves open, but measured flow rate less than $(0.55 + \Delta F)$ (5. 15)

Code 5. Setting Error

$$\text{True}\begin{cases} if & T\_target < T\_\max & or \\ if & T\_target < T\_\min & or \\ if & T\_\max < T\_\min & or \\ & etc. & \end{cases} \tag{5. 16}$$

Where T denotes measured temperature for any channel; ΔT_3, ΔT_4 and ΔT_5 are positive temperature parameters defined by users; ΔF is positive flow rate parameters also defined by users; T-target denotes the maximum temperature that the die insert is heated up in the furnace, T-max is the upper bound of the desired temperature range, and T-min stands for the lower bound as in Section 5.4.

It should be mentioned that code 3-*thermocouple disconnected* has the first priority in the warning module because temperature measurements are the foundation on which the system is built. The flow chart of the warning algorithm for overheating and overcooling is illustrated in Figure 5.18.

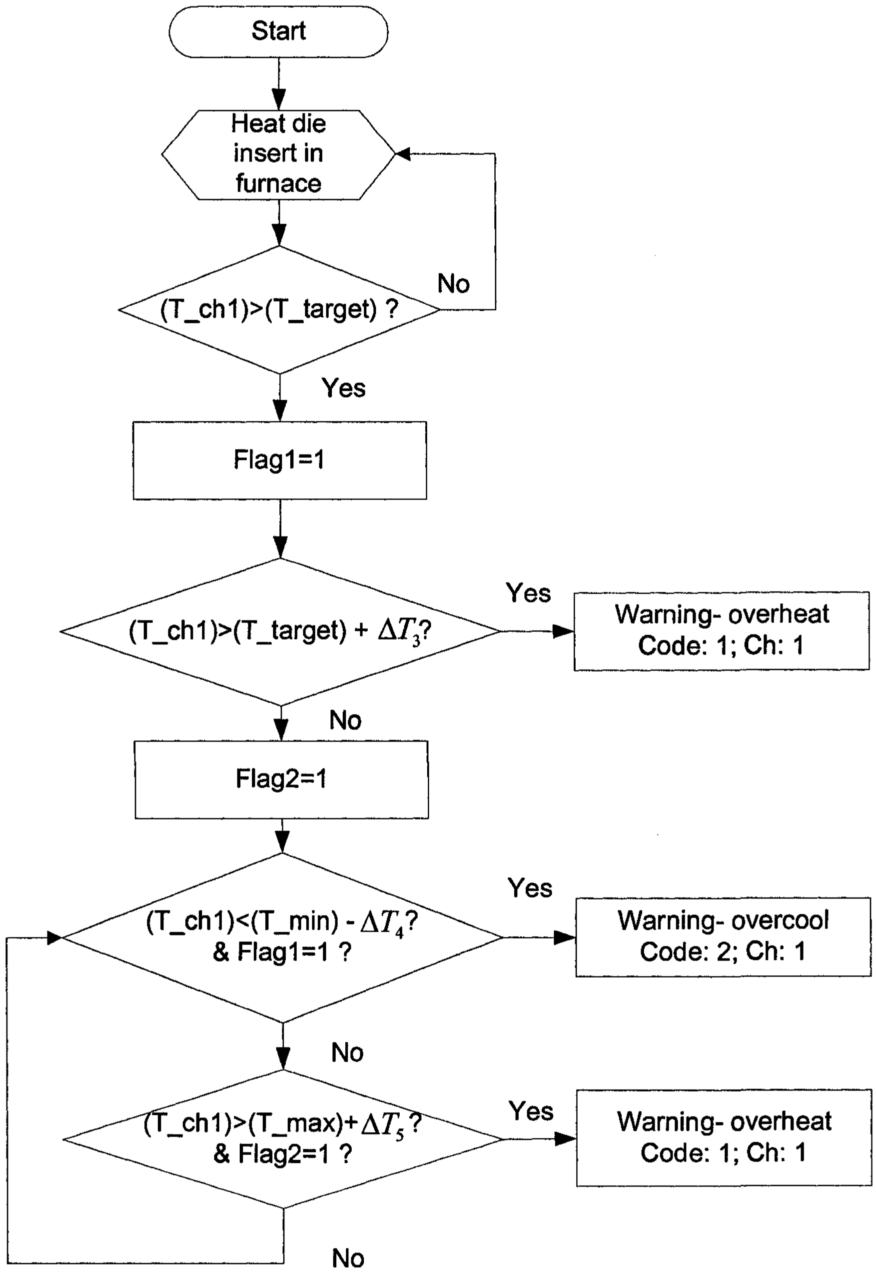


Figure 5.18. Flow Chart of Warning Algorithm for Overheating and Overcooling.

Chapter 6

Conclusions

This chapter summarizes the main conclusions of the present work, and suggestions for future research is also provided in Section 6.2

6.1 Summary of Main Contributions

In this study, a PC-based intelligent real-time monitoring and control system (IRMCS) has been developed for die casting processes where multiple cooling channels are involved. Two controllers have been designed, and numerous experiments have been conducted to verify the performance of the system. The main conclusions can be drawn from above analysis as follows:

It has been demonstrated that local die surface temperature can be deduced from cooling water outlet temperature combined with different flow rates without destructively inserting thermal sensors into a die from its back. A correlation has been established for the die insert used in the experiment.

A new on-line thermal management scheme based on the IRMCS has been proposed. In this scheme, extra cooling water lines controlled by a pump and solenoid valves are hooked up to each established cooling channel. Pump and solenoid valves can be actuated either automatically or manually to introduce additional cooling water to the die insert.

A local temperature controller for multiple cooling channels has been designed to limit die insert temperature in a given range. The performance of the controller has also been evaluated.

A fuzzy controller has been implemented to minimize the occurrence of hot spots in the die insert. The controller is capable of effectively adjusting the pump speed and turn on/off the solenoid valves according to the temperature difference and the error change. Consequently the flow rate of cooling water can be varied to reduce the temperature of the hot spots.

Warning and fault detection functions have been designed and integrated into the developed system. It is able to send aural and visual signals to remind the user of the abnormal conditions detected by the system.

The experimental results obtained from a laboratory die casting simulator indicate that the proposed thermal management scheme and the developed control system are capable of controlling the desired supply of extra cooling water into the dual cooling lines, which effectively controls the local temperature of the die insert.

6.2 Suggestions for Future Research

Numerous improvements need to be carried out in order to put the results of the present work into practical use. The major suggestions for future research are highlighted as follows:

----Die casting simulator. The currently used simulator has limited functions that hinder its ability to simulate complexity involved in real die casting processes. Although it simulates the solidification stage of the die casting process, the stage of die lubrication and the cycling nature of the process cannot be obtained from the current settings. The simulator has to be re-designed to consider the entire cycle of die casting processes, in which die lubrication also reduce the temperature of the die surface and consequently makes contribution to heat transfer from the die to its surroundings.

----Die insert. A casting die often consists of many inserts, and the current system only has one. Therefore, additional inserts are needed to form a complicated cooling system. It is necessary to derive its thermodynamic model for control purpose.

----Temperature sensor. To measure the die insert temperature through thermocouples, holes need to be drilled from the rear of the die insert. There are many disadvantages for using the thermocouple. First, although thermocouple is rugged and inexpensive, the cost of placement is very high. Temperature measurement is sensitive to the thermocouple's location. A small deviation from the desired location because of workmanship can cause a big error in output. Second, it is a kind of sluggish device that needs time to response to a transient process, such as temperature drops in this study. Last but not the least, drilling a hole may upset the maintenance of the die insert, and consequently may reduce its service life. A vision-based thermal management system using an infrared camera needs to be built for further analysis.

Appendix- Main Source Code[1]

```
//-----------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "cbw.h"
#include "Unit1.h"
#include "Unit2.h"
#include <alloc.h>
#include <stdio.h>
#include <math.h>
#include <sysopen.h>
//-----------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

#define BoardNum 1
#define ADRange UNI10VOLTS
#define channels 2
#define samples 250


int i,j,k, pointer, counter, pass1, pass2, error2, delay;
float **flowrate, TRange1, FRange1, TRange2, FRange2, Fmax;
int **temperature,WTarget,WUpper,WLower,Width1,Width2,Height1,Height2;
int TTarget, TControlOn, TControlOff, Tmax, Tmin, pumpspeed, pumpvalue;
int chflag, ch1flag, ch2flag,systemflag;
int pumpflag, pumpflag1,pumpflag2, valve1flag, valve2flag;
int flag, warningflag, warningflag1, warningflag2, warningflag3,
   warningflag4, warningflag5, warningflag6;
TPoint T1point[samples],T2point[samples],T3point[samples],
      T4point[samples],F1point[samples],F2point[samples];
int fuzzytable[5][9]={{750,1500,6000,6000,6000,0,0,-1500,-750},
                      {750,1500,6000,0,0,0,0,-1500,-750},
                      {750,1500,6000,0,0,0,-6000,-1500,-750},
```

[1] This code was written for research purpose only other than commercial use, so it may be not compact and may contain unknown bugs.

```
                {750,1500,0,0,0,0,-6000,-1500,-750},
                {750,1500,0,0,-6000,-6000,-6000,-1500,-750}};
int delaynum=10;
const float Fscalar=0.00108758;
const float Tscalar=0.24439919;
int AverageSize;


TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

//---------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  cbDConfigPort (BoardNum, FIRSTPORTA, DIGITALOUT);
  cbDOut(BoardNum,FIRSTPORTA,0);            //pump control
  cbDConfigPort (BoardNum, FIRSTPORTB, DIGITALOUT);
  cbDOut(BoardNum,FIRSTPORTB,0);            //valve
  cbDConfigPort (BoardNum, FIRSTPORTCL, DIGITALOUT);
  cbDOut(BoardNum,FIRSTPORTCL,0);           //fault channel & alarm
  cbDConfigPort (BoardNum, FIRSTPORTCH, DIGITALOUT);
  cbDOut(BoardNum,FIRSTPORTCH,0);           //fault code

  Timer1->Enabled = false;
  Timer2->Enabled = false;
  ComboBox1->Enabled = false;
  Label40->Visible =false;

  PaintBox1->Visible = true;
  PaintBox2->Visible = true;
  Tmax = 400;
  Tmin = 0;
  Fmax = 3.0;
  flag=0;
  Width1=PaintBox1->Width;
  Height1=PaintBox1->Height;

  Width2=PaintBox2->Width;
  Height2=PaintBox2->Height;

  TRange1 = (float)Width1/(float)samples;
```

```
  TRange2 = (float)Height1/(float)(Tmax-Tmin);
  FRange1 = (float)Width2/(float)samples;
  FRange2 = (float)Height2/(Fmax - 0.5);
  PaintBoxBackground();
  PaintBoxCoordinates();

}

//---------------------------------------------------------------------
void TForm1::PaintBoxBackground(void)2
{
  TRect newRect=Rect(0,0,Width1,Height1);
  PaintBox1->Canvas->Brush->Color=clBlack;
  PaintBox1->Canvas->FillRect(newRect);


  newRect=Rect(0,0,Width2,Height2);
  PaintBox2->Canvas->Brush->Color=clBlack;
  PaintBox2->Canvas->FillRect(newRect);

  PaintBoxCoordinates();
}

//---------------------------------------------------------------------
void TForm1::PaintBoxCoordinates(void)
{
   PaintBox1->Canvas->Pen->Color=clTeal;
   PaintBox2->Canvas->Pen->Color=clTeal;
   for(i=0;i<10;i++)
   {
     PaintBox1->Canvas->MoveTo(i*Width1/10,0);
     PaintBox2->Canvas->MoveTo(i*Width1/10,0);
     PaintBox1->Canvas->LineTo(i*Width2/10,Height);
     PaintBox2->Canvas->LineTo(i*Width2/10,Height);
   }
   for(i=0;i<10;i++)
   {
      PaintBox1->Canvas->MoveTo(0,i*Height1/10);
      PaintBox1->Canvas->LineTo(Width,i*Height1/10);
      PaintBox2->Canvas->MoveTo(0,i*Height2/5);
      PaintBox2->Canvas->LineTo(Width,i*Height2/5);
   }
}

//---------------------------------------------------------------------
```

[2] The curve displaying subroutine was implemented by Ms. Fang Chen in reference [4].

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Memo1->Text = "";
}

//------------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
  temperature=(int **)malloc(4*sizeof(int*));
  for(i=0;i<4;i++)
    *(temperature+i)= (int *)calloc(samples,sizeof(int));

  flowrate=(float **)malloc(channels*sizeof(float*));
  for(i=0;i<channels;i++)
    *(flowrate+i)= (float *)calloc(samples,sizeof(float));

  delay=0;
  pointer=0;
  AverageSize = StrToInt(Form2->ComboBox8->Text);

  systemflag=0;
  Tmax = StrToInt(Form2->ComboBox1->Text);
  Tmin = StrToInt(Form2->ComboBox2->Text);
  int tmp = Tmax-Tmin ;
  if(tmp <= 0 )
     warningflag2 = 1;
  else
  {
    warningflag2 = 0;
    TRange1 = (float)Width1/(float)samples;
    TRange2 = (float)Height1/(float)tmp;
    tmp = tmp/10;
    Label23->Caption = IntToStr(Tmin);
    Label24->Caption = IntToStr(Tmin + tmp);
    Label25->Caption = IntToStr(Tmin + tmp*2);
    Label26->Caption = IntToStr(Tmin + tmp*3);
    Label27->Caption = IntToStr(Tmin + tmp*4);
    Label28->Caption = IntToStr(Tmin + tmp*5);
    Label29->Caption = IntToStr(Tmin + tmp*6);
    Label30->Caption = IntToStr(Tmin + tmp*7);
    Label31->Caption = IntToStr(Tmin + tmp*8);
    Label32->Caption = IntToStr(Tmin + tmp*9);
    Label33->Caption = IntToStr(Tmax);
  }

  float temp= StrToFloat(Form2->ComboBox4->Text);
```

```
 Timer1->Interval = (int)(temp*1000.0);
 Timer1->Enabled = true;
 temp = temp*(float)samples;
 Label19->Caption = IntToStr((int)temp);
 temp = temp/2.0;
 Label21->Caption = IntToStr((int)temp);
 pass1=0;
 pass2=0;
}

//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
 Timer1->Enabled = false;
 Timer2->Enabled = false;
 Label11->Caption= "OFF";
 Label12->Caption= "0";
 cbDOut(BoardNum,FIRSTPORTA,0);
 cbDOut(BoardNum,FIRSTPORTB,0);
 cbDOut(BoardNum,FIRSTPORTCL,0);
 cbDOut(BoardNum,FIRSTPORTCH,0);
}

//---------------------------------------------------------------------------
void TForm1::warning(void)
{
 warningflag = warningflag1|warningflag2|warningflag3|warningflag4|
         warningflag5|warningflag6;
 int tmp=chflag+4;
 if(warningflag==0)
 {
   Label40->Visible = false;
   cbDOut(BoardNum,FIRSTPORTCL,0);
   cbDOut(BoardNum,FIRSTPORTCH,0);
 }
 else
 {
   Label40->Visible = true;
   cbDOut(BoardNum,FIRSTPORTCL,tmp);
 }

 Memo1->Text = "";

 if(warningflag1==1)
 {
   Memo1->Text = Memo1->Text + "\nControl-on upper limit should be "
```

```
              + "greater than control-off lower boundary.\n";
  cbDOut(BoardNum,FIRSTPORTCH,5);
 }
 if(warningflag2==1)
 {
  Memo1->Text = Memo1->Text + "\nDisplay upper limit should be" +
           "greater than lower boundary.\n";
  cbDOut(BoardNum,FIRSTPORTCH,5);
 }
 if(warningflag3==1)
 {
  Memo1->Text = Memo1->Text +"\nChannel " + chflag + " overheat.\n";
  cbDOut(BoardNum,FIRSTPORTCH,1);
 }
 if(warningflag4==1)
 {
  Memo1->Text =Memo1->Text+"\Pump starts, but flow rate low\n"+"\n";
  cbDOut(BoardNum,FIRSTPORTCH,2);
 }
 if(warningflag5==1)
 {
  Memo1->Text = Memo1->Text + "\nChannel " + chflag +
          " thermocouple disconnected.\n";
  cbDOut(BoardNum,FIRSTPORTCH,3);
 }
 if(warningflag6==1)
 {
  Memo1->Text = Memo1->Text+"\nChannel "+chflag +" over-cooled.\n";
  cbDOut(BoardNum,FIRSTPORTCH,4);
 }
}

//-------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 unsigned short T,F;

 int max = StrToInt(ComboBox3->Text);
 int min = StrToInt(ComboBox4->Text);
 if(max <= min )
  warningflag1 =1;
 else
 {
  warningflag1 = 0;
  TControlOn = max;
  TControlOff = min;
```

```
}

TTarget= StrToInt(ComboBox2->Text);
WTarget=TTarget+ StrToInt(ComboBox6->Text);
WUpper=TControlOn+ StrToInt(ComboBox7->Text);
WLower=TControlOff-StrToInt(ComboBox5->Text);

if(RadioGroup1->ItemIndex==4)
  ComboBox1->Enabled = true;
else
  ComboBox1->Enabled = false;

if(RadioGroup1->ItemIndex==1|RadioGroup1->ItemIndex==2|
                            RadioGroup1->ItemIndex==3)
  Button5->Enabled=false;
else
  Button5->Enabled=true;

for(i=0; i<channels; i++)
{
  cbAIn(BoardNum, i, ADRange, &F);
  flowrate[i][pointer] = ((float)F*Fscalar)+0.55;
}
for(i=2; i<6; i++)
{
  cbAIn(BoardNum, i, ADRange, &T);
  temperature[i-2][pointer] = (int)((float)T*Tscalar);
 }
 if(pointer>=AverageSize) systemflag=1;
 if(systemflag==1)
 {
   for(i=0; i<4; i++)
   {
     int tmp=0;
     for(j=0; j<AverageSize; j++)
     {
      k=pointer-j;
      if(k<0) k=samples+k;
      tmp=tmp+temperature[i][k];
     }
     temperature[i][pointer]=int(tmp/AverageSize);
   }
 }

 Label5->Caption = IntToStr(temperature[0][pointer]);
 Label6->Caption = IntToStr(temperature[1][pointer]);
```

```
if(temperature[0][pointer]>5 & temperature[1][pointer]>5)
{
 warningflag5=0;
 chflag=0;
}
if(temperature[0][pointer]<5|temperature[1][pointer]<5)
{
 warningflag5=1;
 if(temperature[0][pointer]<5)
   chflag=1;
 if(temperature[1][pointer]<5)
   chflag=2;
}

if(warningflag5==0)
{
 if(temperature[0][pointer]>WTarget)
 {
   warningflag3=1;
   chflag=1;
 }
 if(temperature[1][pointer]>WTarget)
 {
   warningflag3=1;
   chflag=2;
  }

 if(temperature[1][pointer]<WTarget&temperature[0][pointer]<WTarget
                       & pass1==0 & pass2==0 )
 {
   warningflag3=0;
 }
 if(temperature[0][pointer]>WUpper & pass1==1)
 {
   warningflag3=1;
   chflag=1;
 }
 if(temperature[1][pointer]>WUpper & pass2==1)
 {
   warningflag3=1;
   chflag=2;
 }
 if(temperature[1][pointer]<WUpper & temperature[0][pointer]<WUpper
              & pass1==1 & pass2==1 )
```

```
  {
    warningflag3=0;
  }

  if(temperature[0][pointer]<WLower & pass1==1)
  {
    warningflag6=1;
    chflag=1;
  }
  if(temperature[1][pointer]<WLower & pass2==1)
  {
    warningflag6=1;
    chflag=2;
  }
  if(temperature[0][pointer]>WLower &temperature[1][pointer]>WLower
                          & pass1==1 & pass2==1)
  {
    warningflag6=0;
  }
  if(warningflag3==0 & warningflag6==0)
  {
    chflag=0;
  }
}

Label9->Caption = FloatToStr( flowrate[0][pointer]);
Label10->Caption = FloatToStr( flowrate[1][pointer]);

Control();
if(RadioGroup1->ItemIndex==3)
{
  delay++;
  if(delay>delaynum)
  {
    delay=0;
    Fuzzy();
  }
}
float tmp=StrToFloat(ComboBox8->Text);
if(pumpflag==1 &flowrate[0][pointer]<tmp &flowrate[1][pointer]<tmp)
  warningflag4=1;
else
  warningflag4=0;

if(Button6->Caption=="Alarm On")
  warning();
```

```
  if(Button6->Caption=="Alarm Off")
    cbDOut(BoardNum,FIRSTPORTCL,chflag);

  pointer++;
  if(pointer == samples)
   pointer=0;

  PaintBox();
}

//-------------------------------------------------------------------
void TForm1::PaintBox(void)
{
   int pointer2;
   pointer2 = pointer;

   PaintBox1->Canvas->Pen->Color=clBlack;
   PaintBox1->Canvas->Polyline(EXISTINGARRAY(T1point));
   PaintBox1->Canvas->Polyline(EXISTINGARRAY(T2point));
   PaintBox1->Canvas->Polyline(EXISTINGARRAY(T3point));
   PaintBox1->Canvas->Polyline(EXISTINGARRAY(T4point));


   PaintBox2->Canvas->Pen->Color=clBlack;
   PaintBox2->Canvas->Polyline(EXISTINGARRAY(F1point));
   PaintBox2->Canvas->Polyline(EXISTINGARRAY(F2point));
   PaintBoxCoordinates();

   for (i=0; i<samples; i++)
   {
    T1point[i]=Point(i*TRange1, Height1-(temperature[0][pointer2]-
        Tmin)*TRange2);
    T2point[i]=Point(i*TRange1, Height1-(temperature[1][pointer2]-
        Tmin)*TRange2);
    T3point[i]=Point(i*TRange1, Height1-(temperature[2][pointer2]-
        Tmin)*TRange2);
    T4point[i]=Point(i*TRange1, Height1-(temperature[3][pointer2]-
        Tmin)*TRange2);

    F1point[i]=Point(i*FRange1, Height2-(flowrate[0][pointer2]-
        0.55)*FRange2);
    F2point[i]=Point(i*FRange1, Height2-(flowrate[1][pointer2]-
        0.55)*FRange2);

    pointer2++;
    if(pointer2 == samples) pointer2=0;
```

```
  }

  PaintBox1->Canvas->Pen->Color=clWhite;
  PaintBox1->Canvas->Polyline(EXISTINGARRAY(T1point));
  //PaintBox1->Canvas->Polyline(EXISTINGARRAY(T3point));
  PaintBox2->Canvas->Pen->Color=clWhite;
  PaintBox2->Canvas->Polyline(EXISTINGARRAY(F1point));

  PaintBox1->Canvas->Pen->Color=clYellow;
  PaintBox1->Canvas->Polyline(EXISTINGARRAY(T2point));
  //PaintBox1->Canvas->Polyline(EXISTINGARRAY(T4point));
  PaintBox2->Canvas->Pen->Color=clYellow;
  PaintBox2->Canvas->Polyline(EXISTINGARRAY(F2point));
}
//---------------------------------------------------
void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
 if(pumpvalue==0)
 {
   pumpvalue=1;
   cbDOut(BoardNum,FIRSTPORTA,pumpvalue);
 }
 else
 {
   pumpvalue=0;
   cbDOut(BoardNum,FIRSTPORTA,pumpvalue);
 }
}

//---------------------------------------------------------------------
void TForm1::Control(void)
{
  int tmp;
  tmp = StrToInt(ComboBox9->Text);
  pumpspeed= int(60000/tmp);
  if(RadioGroup1->ItemIndex==2)   //Local Controller, Dual Channels
  {
    RadioGroup2->ItemIndex=2;
    RadioGroup3->ItemIndex=2;
  if(temperature[0][pointer]>(TTarget-1)| temperature[1][pointer]
                                >(TTarget-1))
    {
      Timer2->Interval = pumpspeed;
      Timer2->Enabled = true;
      Label11->Caption="on";
      Label12->Caption = ComboBox9->Text;
```

```
}

if(temperature[0][pointer]>TTarget|temperature[1][pointer]
                                    >TTarget)
{
  flag=1;
  pumpflag=1;
  cbDOut(BoardNum,FIRSTPORTB,3);
  Label7->Caption="ON";
  Label8->Caption="ON";
  Label11->Caption="ON";
}

 if(flag==1)
 {
  if(temperature[0][pointer]<(TControlOff))
  {
   valve1flag=1;
   Label7->Caption="Off";
   pass1=1;
  }

  if(temperature[0][pointer]>(TControlOn))
  {
   valve1flag=0;
   Label7->Caption="ON";

  }

  if(temperature[1][pointer]<(TControlOff))
  {
   valve2flag=1;
   Label8->Caption="Off";
   pass2=1;
  }
  if(temperature[1][pointer]>(TControlOn))
  {
   valve2flag=0;
   Label8->Caption="ON";
  }
  tmp=(1-valve1flag) + 2*(1-valve2flag);
  if(tmp!=0)
  {
   Timer2->Interval = pumpspeed;
   Timer2->Enabled = true;
   Label11->Caption="on";
```

```
        Label12->Caption = ComboBox9->Text;
      }
      cbDOut(BoardNum,FIRSTPORTB,tmp);
    }

      if(valve1flag==1 & valve2flag==1)
       {
        pumpflag=0;
        Timer2->Enabled=false;
        cbDOut(BoardNum,FIRSTPORTA,0);
        cbDOut(BoardNum,FIRSTPORTB,0);
        Label7->Caption="Off";
        Label8->Caption="Off";
        Label11->Caption="Off";
        Label12->Caption ="0";
       }
}

if(RadioGroup1->ItemIndex==1)  //Local Controller, Single Channel
{
  RadioGroup2->ItemIndex=0;
  RadioGroup3->ItemIndex=1;
  if(temperature[1][pointer]>(TTarget-1) )
  {
    Timer2->Interval = pumpspeed;
    Timer2->Enabled = true;
    Label11->Caption="on";
    Label12->Caption = ComboBox9->Text;
  }

  if(temperature[1][pointer]>TTarget )
  {
    flag=1;
    pumpflag=1;
    cbDOut(BoardNum,FIRSTPORTB,2);
    Label7->Caption="Off";
    Label8->Caption="ON";
    Label11->Caption="ON";
   }

   if(flag==1)
   {
    if(temperature[1][pointer]>TControlOn)
     {
      Timer2->Interval = pumpspeed;
      Timer2->Enabled = true;
```

```
            cbDOut(BoardNum,FIRSTPORTB,2);
            Label8->Caption="ON";
            Label11->Caption="on";
            Label12->Caption = ComboBox9->Text;
          }
          if(temperature[1][pointer]<TControlOff)
          {
            Timer2->Enabled = false;
            cbDOut(BoardNum,FIRSTPORTB,0);
            Label8->Caption="Off";
            Label11->Caption="Off";
            pumpflag=0;
            Label12->Caption = "0";
           }
          }
      }
}

//-----------------------------------------------------------------
void TForm1::Fuzzy(void)
{
  int m,n,tmp, u, errorchange;
  float u1,u2,u3,u4,u5;

  tmp=temperature[0][pointer]-temperature[1][pointer];
  errorchange=tmp-error2;
  error2=tmp;
  RadioGroup2->ItemIndex=2;
  RadioGroup3->ItemIndex=2;

/*                          //algorithm 1, direct method
  if(tmp<=-20)
     u=80;
  if(tmp>-20 & tmp<=0)
  {
     u1=(tmp+20.0)/20.0;
     u2=-tmp/20.0;
     u3=0.0;
  }
  if(tmp>0 & tmp<=20)
  {
     u1=-(tmp-20.0)/20.0;
     u2=0.0;
     u3=tmp/20.0;
  }
  if(errorchange<=-4)
```

```
  u4=u1;
if(errorchange>-4 & errorchange<=1)
  u4=min(u1,float((-((float)errorchange+1.0)/3.0)));
else
  u4=0.0;
if(errorchange>=1 & errorchange<4)
  u5=min(u1,float((((float)errorchange-1.0)/3.0)));
if(errorchange>=4)
  u5=u1;
else
  u5=0.0;

u=int(((u2-u3)*80.0+(u4-u5)*40.0)/(u1+u2+u3+u4+u5));
if(u>=40)        u=80;              //quantized
if(u>15 & u<40)    u=40;
if(u>5 & u<=15)    u=10;
if(u>=-5 & u<=5)   u=0;
if(u>-15 & u<-5)   u=-10;
if(u>-40 & u<=-15) u=-40;
if(u<=-40)         u=-80;
if(u!=0)
{
  Timer2->Interval = abs(60000/u);
  Label11->Caption="On";
  Label12->Caption = IntToStr(abs(u));
  if(Timer2->Enabled == false)
     Timer2->Enabled = true;
  if(u<0)
  {
    cbDOut(BoardNum,FIRSTPORTB,1);
    Label7->Caption="On";
    Label8->Caption="Off";
  }
  if(u>0)
  {
    cbDOut(BoardNum,FIRSTPORTB,2);
    Label7->Caption="Off";
    Label8->Caption="On";
  }
}
else
{
  Timer2->Enabled = false;
  cbDOut(BoardNum,FIRSTPORTA,0);
  cbDOut(BoardNum,FIRSTPORTB,0);
  Label7->Caption="Off";
```

```
    Label8->Caption="Off";
    Label11->Caption="Off";
    Label12->Caption = "0";
  }
*/

/*                              //algorithm 2, look-up table
  if(tmp<-10)
  {
    m=0; n=0;
  }
  if(tmp>-10 & tmp<-3)
  {
    m=0; n=1;
  }
  if(tmp==-2)
  {
    n=2;
    if(errorchange<1)
      m=0;
    else
      m=3;
  }
  if(tmp==-1)
  {
    n=3;
    if(errorchange==-2)
      m=0;
    else
      m=1;
  }
  if(tmp==0)
  {
    n=4;
    if(errorchange==-2)
      m=0;
    if(errorchange==2)
      m=4;
    else
      m=1;
  }
  if(tmp==1)
  {
    n=5;
    if(errorchange==2)
      m=4;
```

```
    else
      m=0;
  }
  if(tmp==2)
  {
    n=6;
    if(errorchange>-1)
      m=2;
    else
      m=0;
  }
  if(tmp>3 & tmp<10)
  {
    m=0;
    n=7;
  }
  if(tmp<-10)
  {
    m=0;
    n=8;
  }
  tmp= fuzzytable[m][n];
  Timer2->Interval = abs(tmp);
  Label11->Caption="On";

  if(Timer2->Enabled == false)
    Timer2->Enabled = true;
  if(tmp>0)
  {
    cbDOut(BoardNum,FIRSTPORTB,1);
    Label12->Caption = IntToStr(60000/tmp);
    Label7->Caption="On";
    Label8->Caption="Off";
  }
  if(tmp<0)
  {
    cbDOut(BoardNum,FIRSTPORTB,2);
    Label12->Caption = IntToStr(-60000/tmp);
    Label7->Caption="Off";
    Label8->Caption="On";
  }
  else
  {
    Timer2->Enabled = false;
    cbDOut(BoardNum,FIRSTPORTA,0);
    cbDOut(BoardNum,FIRSTPORTB,0);
```

```
    Label7->Caption="Off";
    Label8->Caption="Off";
    Label11->Caption="Off";
    Label12->Caption = "0";
  }
*/

}

//---------------------------------------------------------------------------
void __fastcall TForm1::Save1Click(TObject *Sender)
{
 int pointer2, tmp;
 char filename[60];
 FILE *data;

 if(SaveDialog1->Execute())
 {
     sprintf(filename, "%s", SaveDialog1->FileName);
     if((data = fopen(filename,"w"))==NULL)
     {
       Application->MessageBox("Could not open a file", "error",
                                              MB_OK);
        return;
     }

     for(i=0;i<channels;i++)
     {
        pointer2 = pointer;
        fprintf(data,"\nFlow Rate: CHANNEL %d: ",i+1);
        for(j=0;j<samples;j++)
        {
           tmp=int(flowrate[i][pointer2]*100);
           if(temperature[i][pointer2]>99)
                fprintf(data,"%d ",tmp);
           else
                fprintf(data,"%d  ",tmp);
           pointer2++;
           if(pointer2>samples)
             pointer2 = 0;
        }
      }

      for(i=2; i<6; i++)
      {
         pointer2 = pointer;
```

```
        fprintf(data,"\nTemperature: CHANNEL %d: ",i-channels+1);
        for(j=0;j<samples;j++)
        {
          if(temperature[i-2][pointer2]>99)
               fprintf(data,"%d ",temperature[i-2][pointer2]);
          else
               fprintf(data,"%d  ",temperature[i-2][pointer2]);
          pointer2++;
          if(pointer2>samples)
            pointer2 = 0;
          }
      }
    fclose(data);
   }
}

//-----------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
  int bitvalue=0;
  flag=0;
  pumpflag = 0;

  if(RadioGroup2->ItemIndex ==0)
  {
    pumpflag1=0;
    Label7->Caption="Off";
  }
  if(RadioGroup2->ItemIndex ==1)
  {
    pumpflag1=1;
    Label7->Caption="On";
  }
  if(RadioGroup3->ItemIndex ==0)
  {
    pumpflag2=0;
    Label8->Caption="Off";
  }
  if(RadioGroup3->ItemIndex ==1)
  {
    pumpflag2=1;
    Label8->Caption="On";
  }
  if(RadioGroup1->ItemIndex==0)
  {
    cbDOut(BoardNum,FIRSTPORTA,0);
```

```
    cbDOut(BoardNum,FIRSTPORTB,0);
    cbDOut(BoardNum,FIRSTPORTCL,0);
    cbDOut(BoardNum,FIRSTPORTCH,0);
    RadioGroup2->ItemIndex=0;
    RadioGroup3->ItemIndex=0;
    Label8->Caption="Off";
    Label7->Caption="Off";
    Label11->Caption= "OFF";
    Label12->Caption= "0";
    Timer2->Enabled = false;
    systemflag=0;
    pass1=0;
    pass2=0;
    warningflag=0; warningflag1=0;warningflag2=0;
    warningflag3=0;warningflag4=0;warningflag5=0;
     //warningflag6=0;
  }

  if(RadioGroup1->ItemIndex==4)
  {
     pumpflag = pumpflag1|pumpflag2;
     bitvalue = pumpflag1 + 2*pumpflag2;
     cbDOut(BoardNum,FIRSTPORTB,bitvalue);
     pumpspeed= StrToInt(ComboBox1->Text);
     if(pumpspeed!=0)
     {
        Timer2->Interval = (int)(60000/pumpspeed);
        Timer2->Enabled = true;
        pumpflag=1;
        Label11->Caption= "On";
        Label12->Caption= ComboBox1->Text;
     }
     else
     {
         cbDOut(BoardNum,FIRSTPORTA,0);
         Label11->Caption= "On";
         Label12->Caption= "0";
         Timer2->Enabled = false;
     }
   }
}

//-------------------------------------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
 if(Button6->Caption=="Alarm On")
```

```
   Button6->Caption="Alarm Off";
 else
   Button6->Caption="Alarm On";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Settings1Click(TObject *Sender)
{
 Form2->ShowModal();
}
//---------------------------------------------------------------------------
```

Reference

[1] North American Die Casting Association (NADCA), "Introduction to die casting," http://www.diecasting.org , Feb. 2005.

[2] H. Hu, F. Chen, X. Chen, Y. Chu, and P. Cheng, "Effect of cooling water flow rates on local temperatures and heat transfer of casting dies," Journal of Materials Processing Technology, Vol. 148, Jan 2004, pp. 439–451.

[3] ----, "Development of a computer-based data acquisition and control system for die casting Processes," Proceedings of the EPD Congress 2003, TMS, California, 2002, pp. 439–451.

[4] F. Chen, "Data acquisition system for die casting processes," M.A.Sc. Thesis, University of Windsor, Canada, 2002.

[5] W.D. Kaiser, S.D. Sanders, and P.D. Frost, "The Ilzro-Battelle multichannel temperature controller," Proceedings of the Seventh SDCE International Die Casting Congress, NADCA, 1972, p. 5172.

[6] B.K. Dent, and R. Fifer, "Production operation with the Ilzro-Battelle die temperature controller," Proceedings of the Seventh SDCE International Die Casting Congress, NADCA, 1972, p. 5572.

[7] S.E. Booth, "A die temperature cycle time controller," Proceedings of the Sixth SDCE International Die Casting Congress, NADCA, 1970, p. 54,

[8] A.P. Peterson, "Thermocycling control of aluminum die casting machines," Proceedings of the Eighth SDCE International Die Casting Congress, NADCA, 1975, B-t75–024,

[9] W. Bishenden, and R. Bhoia, "Die temperature control," Proceedings of the Transactions of 20th International Die Casting Congress, NADCA, Cleveland, OH, 1999, pp. 161–164.

[10] P. Misra, "System modeling," T. Samad (ed.) Perspectives in Control Engineering, IEEE Press, 2001, pp. 71-103.

[11] L. H. Tsoukalas and R. E. Uhrig, "Fuzzy and neural approaches in Engineering," Jonh Wiley & Sons, 1996.

[12] Eric A. Wan, "Control Systems: Classical, Neural, and Fuzzy," Oregon Graduate Institute, Lecture Notes, 1998.

[13] J. S. Roger Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference Systems," IEEE Transaction on System, Man, and Cybernetics, Vol. 23, No. 3, 1993, pp665-685.

[14] ----, "Neuro-Fuzzy and Soft Computing," Prentice Hall, 1997

[15] L. A. Zadeh, "Fuzzy Sets," Information and Control, Vol. 8, pp338-353, 1965.

[16] T. Takagi and M. Sugeno, "Derivation of Fuzzy Control Rules from Human Operator's Control Action," Conference on Fuzzy Information Knowledge Representation and Decision Analysis, 1983, pp55-66.

[17] ----, "Fuzzy Identification of Systems and its Application to Modeling and Control," IEEE Transaction on System, Man, and Cybernetics, Vol. 15, 1985, pp116-132.

[18] Chuen Chien Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I," IEEE Transaction on System, Man, and Cybernetics, Vol. 20, 1990, pp401-415.

[19] ----, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II," IEEE Transaction on System, Man, and Cybernetics, Vol. 20, 1990, pp419-433.

[20] M. M. Gupta, and D. H. Rao (ed.), "Neuro-Control Systems: Theory and Applications," IEEE Press, 1994.

[21] MATLAB, "Fuzzy Logic Toolbox," Mathwork Inc., 2005

[22] Cyberresearch Inc., RCP3212DA user manual, http://www.cyberresearch.com , Feb. 2005,

[23] Omega Inc., Datasheet of thermocouple HUKMQSS and valve SV-301, http://www.omega.com/ , Feb. 2005.

[24] Analog Device Inc., Datasheet of AD595, http://www.analog.com , Feb. 2005.

[25] GEMS Sensors Inc., Datasheet of RFA-type Sensor, http://www.gemssensors.com , Feb. 2005.

[26] Wilden Pump & Engineering Co., User Manual of Accu-Flo Pump, Feb. 2005.

[27] Omega Inc., Datasheet of valve SV-301, http://www.omega.com/ , Feb. 2005.

[28] X. Niu, J. Kiriakakis and K. Steinbauer, "Application of Infrared Thermal Imaging Technique in High Pressure Die Casting," NADCA, pp101-108, 2003, T03-023

[29] S. Bounds, K. Davey, and S. Hinduja, "An Experimental and Numerical Investigation into the Thermal Behavior of the Pressure Die Casting Process," Transactions of the ASME, Vol. 122, 2000, pp90-99.

[30] S. J. Vargo, A.B. Draper, and J. K. Sprinke, "Waterline Location within a Die Casting Die," 13th SDCE International Die Casting Congress, NADCA, 1985, G-T85-066

[31] Shou-Shing Hsieh, "Transient Thermal Analysis of the Waterlines Effect in the die Casting Dies," Appl. Math. Modeling, Vol. 13, 1989, pp282-289,

[32] Jonathan Papai and Carroll Mobley, "Die Thermal Fields and Heat Fluxes During Die Casting of 380 Aluminum Alloy in H-13 Steel Dies," NADCA, 1991, T91-OC1.

[33] H. Hu, "Casting: Modeling and Simulation," Lecture Notes for Course 89-511, University of Windsor, 2005.

[34] J. Campbell, "Casting," Butterworth Heinemann, Oxford,1991

[35] D. R. Poirier, and G. H. Geiger, "Transport Phenomena in Materials Processing," John Wiley & Sons, 1960

[36] F. P. Incropera, and D. P. Dewitt, "Introduction to Heat Transfer," John Wiley & Sons, 1990

[37] L. C. Thomas, "Heat Transfer – Professional Version," Capstone Publishing Corporation, 1999

[38] T. Yang, H. Hu, X. Chen, Y. Chu, and P. Cheng, "Thermal Analysis of a Die Cooling System," NADCA, 2005, T05-132.

[39] ----, "On-line Thermal Management System for Die Casting Processes," Proc. 5th International Workshop on Advanced Manufacturing Technologies, London, ON, Canada, May 2005

[40] ----, "An Intelligent Control System for Die Thermal Management," Proc. Automation & Control/Advanced Metallurgical Models/Sensors, Materials Science & Technology 2005, TMS, Pittsburg, PA, USA, Sept. 2005 (Accepted).

Vita Auctoris

Tiebao Yang was born on September 22, 1972 in Henan Province, PR China. He received his Bachelor of Engineering degree in Electrical Engineering from Civil Aviation University of China in July 1993. He has worked as an electrical engineer for General Electric (GE), Shenzhen Airlines and Beijing Airlines since graduation. He enrolled at the University of Windsor in January 2004, and graduated with a Master of Applied Science degree in Electrical and Computer Engineering in May 2005. He is now accepted as a doctoral student in Electrical and Computer Engineering at the University of Windsor.