

# **Household Climate Control System**

**A.k.a. WIN2K1**

Wai Nang Tsang - 660065382

Thomas Lo - 660163469

Tan Cheung - 660064411

Michael Lee - 660063111

Section 1, 2(Tan)  
Dec 11<sup>th</sup>, 2001



**Rensselaer**

# Table of Contents

---

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>1-3</b>
1.1	Project Summary.....	1-3
1.2	Problem Statement.....	1-3
<b>2.</b>	<b>IMPLEMENTATIONS.....</b>	<b>2-4</b>
2.1	Materials and Methods .....	2-4
2.1.1	Materials .....	2-4
2.1.2	LCD Display .....	2-4
2.1.3	Stepper Motor .....	2-6
2.1.4	Temperature Sensors.....	2-7
2.1.5	Fuzzy Logic .....	2-8
2.2	Result/Discussion .....	2-11
2.2.1	LCD Display .....	2-11
2.2.2	Stepper Motor .....	2-11
2.2.3	Temperature Sensors.....	2-11
2.2.4	Fuzzy Logic .....	2-12
<b>3.</b>	<b>BACK MATERIALS.....</b>	<b>3-13</b>
3.1	References .....	3-13
3.2	Bibliography.....	3-13
3.2.1	URLs.....	3-13
3.2.2	Printed Materials.....	3-13
3.3	Appendices .....	3-14
3.3.1	Circuit diagrams.....	3-14
3.3.2	Flow Chart .....	3-18
3.3.3	Codes .....	3-19

FIGURE 2: LM34DZ BOTTOM VIEW .....	2-8
FIGURE 3: OP-AMP CIRCUIT FOR SENSOR .....	3-14
FIGURE 4: KEYPAD CIRCUIT .....	3-15
FIGURE 5: LEDS CIRCUIT (HEAT LIGHT) .....	3-16
FIGURE 6: LCD DISPLAY .....	3-16
FIGURE 7: FAN CIRCUIT .....	3-17

## List of Tables

---

TABLE 1: STEPPER MOTOR TABLE.....	2-7
TABLE 2: FUZZY LOGIC RULE MATRIX .....	2-9
TABLE 3: REFERENCES .....	3-13
TABLE 4: FUZZYWUZZY.C.....	3-28
TABLE 5: KEYPAD.H .....	3-29

## Abstract

---

Our team's WIN2K1 is basically a household climate control system that allows user to keep their home environment as they wish. Money saving and energy conservation is our motivation to design this project.

# 1. Introduction

---

## 1.1 Project Summary

With the ever-increasing climate control systems in households, we feel that these systems are inefficient with energy conservation. We wish to improve upon current design, to improve functionality resulting in better energy efficiency and ease of use. We see opportunity in improving upon these systems by adding more enhancements, i.e. fuzzy logic, user-friendly LCD interface. Our unit will open or close windows when the temperature is too hot, warm or too cold. We checked and maintained the temperature by using A/D conversion and interrupts. We used a sensor to monitor both the current room temperature and the outside temperature for accuracy of our fuzzy logic cases. Our fuzzy logic will determine trends in temperature and operate in winter, summer and spring/fall mode.

## 1.2 Problem Statement

There is a need for our product because of the significant impact of California energy crisis and the dwindling supply of fossil fuel. This problem affects the general public and will deplete the supply of energy for future generations. As a result, by using our products, homeowners will come to appreciate the savings in energy costs.

## 2. Implementations

---

This section will go through the details about different parts of our project; materials and methods, results and discussion ~~will be included in this section.~~

### 2.1 Materials and Methods

#### 2.1.1 Materials

Our project contains various parts:

- 2 temperature-sensing units (LM34DZ)
- 1 LCD display
- 1 keypad
- 1 12V stepper motor with IC chip package
- 2 LEDs
- 1 fan
- 2 transistors
- 2 LM301A / LM201A

#### 2.1.2 LCD Display

In our project, we ~~will use~~ the LCD display to output the menu and a keypad to get user inputs. In the menu, we have 4 functions: Open window, close window, Smart Mode and display current temperature. Basically, the LCD display and the keypad is just an interface between the user and the HC12. It allows the user to gain full control of the system. In the open and close mode, our program will prompt the user to enter a value to determine how much window is going to be open or closed. Once the user entered the SMART mode, the LCD panel will temporarily go into a stage that user don't have any access to it, until the '#' key is pressed, which means the end of the smart mode. Lastly, when the users choose the display temperature mode, the LCD will display the current room temperature on the LCD screen.

In order to make our LCD display working; we do this by using all the functions in the provided header file call LCD12.h. This header file includes all the functions, which are required to initialize the LCD display and to write data to it. The OpenXLCD function is used to initialize the LCD before it could be used. The SetDDRamAddr function is used to determine which position the data will be displayed on the LCD. The BusyXLCD function is used to check whether or not the LCD is currently doing something. If it is, then the program should wait for the process to be done before doing anything else with the LCD. The WriteCmdXLCD function is used to write a command to the LCD controller. It basically uses the WriteDataXLCD and WriteBuffer functions to write characters to the LCD.

Beside the header file, we also had to write some C program to make use of the header file. Since we are displaying the menu and the temperature value to the LCD screen, we have to store all

these value as strings for later use. All the items of the menu are stored as individual strings. The temperature value we will display on the screen also stored as string. The only thing we need to do is to obtain the temperature value from the temperature sensors' functions. When outputting the data to the screen, we first have to clear the display using the WriteCmdCLCD function and passing in a value of 0x01 to the eight bits of the LCD controller. We then set the starting position of the LCD display at the upper left-hand corner to write the four choices of the menu. The second item is on the upper right-hand corner, etc. We do it by passing in the address of the location into the SetDDRamAddr function. The locations with their corresponding addresses can be obtained in Figure 4 of the supplement "Interfacing a Hitachi HD44780 to a Motorola 68HC12" at the end of the 68HC12 user's manual.

In order for us to have the keypad working, we wrote a header file called keypad.h, which is used to initialize the Key Wakeup Interrupt and contains the code for the Interrupt Service Routine. In our program, we have to use two global variables. The main program will also recognize these variables and is a way of transferring data from the results obtained in the header file to the main program. We first initialized Port J to do a variety of things. Port J must be set to output from bits 4 to 7 and input from bits 0 – 3. Bits 4 to 7 are where the row select of the keypad are connected to and bits 0 to 3 are where the column selects are connected. All the row selects are pulled low and the column selects are pulled high normally. It will be pulled low when the button on the keypad is pressed and the switch closes. Next, falling edges have to be detected to set a flag. Also, Port J must be configured to enable pull-ups. Pull-ups are then selected for bits 0 to 3 while bits 4 to 7 are set to low to allow for grounding of the row selects. Then, keypad interrupts are enabled for bit 0 to 3.

The Key Wakeup ISR must do seven things. It must save the state of the Key Wakeup Flags. It must then determine which column the key pressed is in. Since the column select go low when pressed, we check to see which columns has been set low by ANDing the value on Port J with 0x08 first. If pin 3 is 0, then the result will be 0 also and the result of the conditional would be true meaning that the keypad is in column three. If bit 3 is 1 meaning that column 3 is not active, then the ANDing of Port J with 0x80 will return a 1 and the conditional will return a 0 meaning that the key pressed is not in column 3. We continue doing this for the other columns until the answer is obtained.

The hardest thing is to check for the row. When a key is pressed the value of the low bits determines which column was activated. Since the keypad we are using does not have a common ground for all the keys, the ground that is seen on the low bit of the Port J is passed in through the row select bits. In order to determine which row is active the program must write a high to each row individually and then look at the value of the low nibble of Port J. If the row you wrote a one to is the row in which the button is pressed then the value on the low bits of Port J will be (- - - - 1111). The function needs to write a one to each row and look for the (1111). When the (1111) is detected it will then look at the original value of the low bits to determine which key was activated.

The next thing that the header file must do is store the ~~inputted~~ value of the keyboard as another global variable, which needs to be used later in the main program. This final value is obtained from a two-dimensional array, which was defined earlier storing all the characters of the keypad into its associated column and row. Afterwards, Port J is reset to all lows and all the

interrupt flags are reset. The Keyflag must also be enabled indicating the button has been released. For more details on the keypad header file, please refer to Appendix.

Now we need to implement these 2 pieces of codes together. After we have initialized the counter and the LCD we now also have to initialize Port J using the function InitPortJ from the header file. Inside the infinite loop, the function getKeyPadEntry is used to get input from the keypad. This function makes use of the KeyFlag and Input global variables from the keypad header file. We then get the user inputted value to determine what the user want to do. We do this by constructing a switch loop as the following:

```
Switch(inString[0])
{
    case '1':    OpenMode();
    Break;
    case '2':    CloseMode();
    Break;
    case '3':    SmartMode();
    Break;
    case '4':    GetTemp();
    Break;
    default:
    break;
}
```

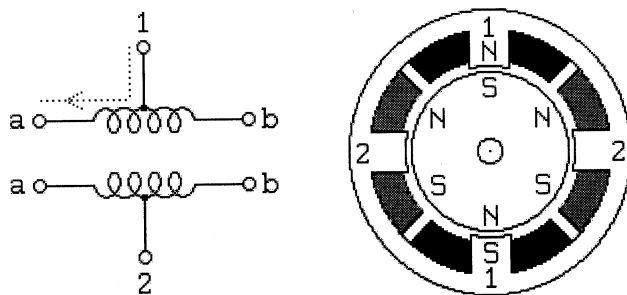
In our program, we constructed 2 functions call write2LCD and writenum2LCD. Write2LCD is a printing function that will only handle strings, and the writenum2LCD will only handle integers. In the GetTemp() function, we have used the writenum2LCD function in order to output the integer value we obtained from the A/D conversion to the LCD screen.

### 2.1.3 Stepper Motor

As a brief introduction to stepper motor, there are two kinds of stepper motor, *permanent magnet* and *variable reluctance*. Permanent magnet motors tend to "cog" as you twist the rotor with your fingers, while variable reluctance motors almost spin freely. Stepping motors come in a wide range of angular resolution. The coarsest motors typically turn 90 degrees per step, while high-resolution permanent magnet motors are commonly able to handle 1.8 or even 0.72 degrees per step, while the one we used in this project can handle 3.6 degrees per step. For both permanent magnet and variable reluctance stepping motors, if just one winding of the motor is energized, the rotor will snap to a fixed angle and then hold that angle until the torque exceeds the holding torque of the motor, at which point, the rotor will turn, trying to hold at each successive equilibrium point.

The motor we used in this project variable reluctance. In order to spin the motors, we have to send in control signals to open and close the windings in the motor at the appropriate times. We therefore use our MC68HC12 Port G to handle this. Although a basic stepper motor has 4 magnetic windings for turning the motor, we only used two of them due to the limitation of ports in the EVB. The schematic is shown in Figure-8. Since the motor requires 12V inputs, as seen in the schematic, we used an integrated circuit (ULN2803) that containing Darlington arrays which we could supply a

5V to the IC and produce a 12V on the output. In order for the ULN2803 to work correctly, we have to supply a 12V on pin 10 and pull pin 9 to ground. Then we connect the EVB J8 pin 15 and 16 to the ULN2803 pin 4 and pin 3 accordingly. And for the ULN2803, we also need to feed back the output of pin 13 and 18 to pin 5 and 2 so that the signal to the input will be inverted. These four signals will be used to drive the motor. Afterwards, we connected the four wires (red, brown, green, white), which represent the coils 1, 3, 2 and 4, from the motor to the ULN2803 pin 17,16,15 and 14 respectively.



**Figure 1: 2 coils Stepper Motor**

In the software point of view, we first have to set the data direction of Port G to output (all bits to 1) by, `_H12DDRG = 0xFF`. Then, we send in signals to PORT G pin 0 and 1 in the following order,

	Open (anti-clockwise)				Close (clockwise)			
<b>PG 0</b>	0	1	1	0	0	0	1	1
<b>PG 1</b>	0	0	1	1	0	1	1	0

**Table 1: Stepper Motor Table**

In any interval of the above sequence, we put a for-loop in between just to make sure the motor has enough time to turn. It is done as follow

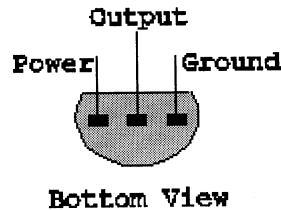
```
_H12PORTG = 0x0F & 0x00;
for ( i=0; i<2000;i++);
_H12PORTG = 0x0F & 0x02;
for (i=0;i<2000;i++);
_H12PORTG = 0x0F & 0x03;
for (i=0;i<2000;i++);
_H12PORTG = 0x0F & 0x01;
for (i=0;i<2000;i++);
```

If you want to change the direction of the spinning, just change the sequence of setting port G as mentioned before.

## 2.1.4 Temperature Sensors



For the temperature-sensing units, we used a LM34DZ thermal transistor. It needs a +5V power supply, and outputs a voltage linearly proportional to the temperature. From the datasheets provided, 1 degree will result in a .01V output. For example, a temperature reading of 75 degrees Fahrenheit, the output will be .75V and at 65 degrees Fahrenheit, the output is .65V.



*Figure 2: LM34DZ Bottom View*

For our purposes, this voltage is too minuscule to work with. In order to get a better reading, the voltage needs to be amplified. We used an op-amp for this purpose. By using an operational amplifier in a non-inverting configuration with a gain of 5, we can scale the output from the thermal transistor to a value that's more suitable for the A/D converter on the M68HC12. For more details on the configuration of the op-amp, a schematic has been attached at the end of this report.

We use the A/D converter to constantly take readings on its channels. For this reason, we configure the A/D converter with the MULT and SCAN bits on. We also enable the converter to generate an interrupt whenever, it's completed obtaining data since this is an interactive program - we don't want to constantly poll the A/D converter and waste precious cycles just waiting for data. We don't need to obtain measurements within short periods of time, thus we set the A/D converter to take samples using a period of 16 cycles, and adjust the P-clock so that the A/D receives a clock frequency with the allowable range. We must also assign the Interrupt jump vector to the ISR so that the interrupt is handled correctly.

```
DB12->SetUserVector(9,ADInt);

_H12ADTCTL2 = 0xC2;      //power up ATD
for ( i =0; i < 1000; i++); //wait for ATD to power up
_H12ADTCTL4 = 0x61;      //set ATD clock and P-clock
_H12ADTCTL5 = 0x73;      //set MULT =1 for multi-channel operation
```

Now, in the ISR routine, the temperature readings on the outside and inside are updated.

```
__mod2__ void ADInt()
{
    outside_temp = _H12ADR2H;
    inside_temp = _H12ADR3H;
    _H12ADTCTL5 = 0x73;
}
```

## 2.1.5 Fuzzy Logic

The core idea of our climate control system was to conserve energy. The only way to do that is to have the system sense differences in temperature inside and outside and adjust the cooling and

heating appropriately so that we can use fuel to its full potential and not waste energy. Every possible combination of scenarios must be identified to ensure that energy used for heating and cooling is not wasted. We turned to the concept of fuzzy logic. It allows us to attach an unambiguous numerical meaning to linguistic term such as “warm” and “cool”. There are several processes to do fuzzy logic for our system. We must take input from the temperatures sensors that have been through A/D conversion and fuzzify it. We put it through fuzzification to get it in input that we can use. We then place the results from the fuzzification into our rule matrix and defuzzify it into states for our system. Our system controls the degree of the opened window the degree of the heating system and cooling system. Our rule matrix is based on differences on the outside temperature and inside temperature. Based on our AD conversions we get readings from 40 to 230. After some extensive testing we determined that below 90 is relatively cold and about 180 is hot. Between 90 and 135 is “cool” and between 135 and 180 is “warm”.

Below is our fuzzify function where we translate a incoming temperature to a value we can use. The returned value of 0,1,2 and 3 are form the 4 by 4 rule matrix.

```
int Fuzzify(int temp)
{
    if (temp >= 180)
        return 3;
    if ((temp > 134) && (temp < 180))
        return 2;
    if ((temp > 90) && (temp < 135))
        return 1;
    if (temp <= 90)
        return 0;
}
```

Here is the rule matrix by this configuration we defuzzify the result and tell the system to change states accordingly.

	<b>Under 15°c</b> <b>[0]</b>	<b>15°c - 23°c</b> <b>[1]</b>	<b>23°c - 30°c</b> <b>[2]</b>	<b>Above 30°c</b> <b>[3]</b>
<b>Below 15°</b> <b>0</b>	Fan: Off Heat: High Window: Closed <i>State 1</i>	Fan: Off Heat: Low Window: Closed <i>State 2</i>	Fan: Off Heat: Off Window: Closed <i>State 3</i>	Fan: Low Heat: Off Window: Closed <i>State 4</i>
<b>15°</b> <b> </b> <b>23°</b> <b>1</b>	Fan: Off Heat: Low Window: Half <i>State 5</i>	Fan: Off Heat: Off Window: Half <i>State 6</i>	Fan: Low Heat: Off Window: Half <i>State 7</i>	Fan: High Heat: Off Window: Half <i>State 8</i>
<b>23°</b> <b> </b> <b>30°</b> <b>2</b>	Fan: Off Heat: Off Window: Half <i>State 9</i>	Fan: Low Heat: Off Window: Full <i>State 10</i>	Fan: Low Heat: Off Window: Full <i>State 10</i>	Fan: High Heat: Off Window: Full <i>State 11</i>
<b>30°+</b> <b>3</b>	Fan: Off Heat: Off Window: Full	Fan: Low Heat: Off Window: Half	Fan: Low Heat: Off Window: Closed	Fan: High Heat: Off Window: Full

	State 12	State 7	State 13	State 14

**Table 2: Fuzzy Logic Rule Matrix**

By our calculations for the system we developed there are 14 different states the window, heater and cooling system can in, however for a bigger model as a real house there will be a lot more states. We insert the desired states in a 4 by 4 matrix and when we apply the fuzzy logic rules we will get the one most energy efficient state for that temperature. Below is the fuzz rule for our system.

```
int FL(int tempout, int tempin)
{
    int fuzzrule[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,10,11},{12,7,13,14}};
    return fuzzrule [tempout][tempin];
}
```

In our main program we run a smart mode function in a loop until a touch of the keypad which disengages the smart mode. And returns to user controlled mode. We assume that most of the time the system should be in smart mode.

```
void SmartMode(void)
{
    int fuzzytemp1, fuzzytemp2, I, rule, movewindow;
    char stop= '';
    stop = Input;
    smartmode =1;
    while ( stop != '#')
    {
        stop = Input;
        write2LCD("In Smart Mode");
        DB12->printf("\rIn Smart Mode\r\n");
        if(1)
        {
            inside_temp/7);
            DB12->printf("%d\r\nOutside temp: %d\r\n", outside_temp/7,
            outside_temp/7);
            DB12->printf("%d\r\nWindow Position: %d\r\n", loc,loc);

            fuzzytemp1=Fuzzify(outside_temp);
            fuzzytemp2=Fuzzify(inside_temp);
            rule = FL(fuzzytemp1, fuzzytemp2);
            switch (rule){
                case 1: _H12PORTS=0x80;//high heat,both in/outside cold
                    DB12->printf("In case 1\r\n\r\n");
                    if (loc < 5)
                        close_window(5-loc);
                    break;
                case 2: _H12PORTS=0x40;//low heat,out cold,in cool
                    DB12->printf("In case 2\r\n\r\n");
                    if (loc < 5)
                        close_window(5-loc);
                    break; //and so on for all 14 states
            }
        }
    }
}
```

We use the case statement above to select the different mode of our system. Port S on the HC12's EVB controls the LEDs, which are to represent our heating system and the fan, which has two modes a high mode and a low mode.

The LEDs are connected very simply to the Port S, pin 6 and 7. When it is on low heat pin 6 will light up and when it is on high heat pin 7 will light up. For cooling system we choose to use a computer-cooling fan to represent the air-cooling system of a house. Since the EVB did not supply enough current to power the fan we must use +5V and +12V to power the fan in the 2 modes. We use transistors to do this. Signal from port S, pin 4 and 5 are used to activate the transistors and allow the current to pass. One transistor is connected to the +5V there the fan will be on low. The other transistor is connected to +12V which when activated will produce high fan.

## **2.2 Result/Discussion**

### **2.2.1 LCD Display**

We have successfully display our desire menu to the LCD screen; users are able to enter their choices without any problem such as the key is not responding on the keypad. And correct functions were called with the correct button being pushed. The display temperature function is working perfectly, and the LCD output the exact number we obtain from the A/D conversion. If we have more time to work on this project, we were thinking to implement a better LCD screen, which will have better visual effects for users. We also think that using the touch screen monitor, as our LCD screen is one of the enhancements we can do to improve our project design and functionality.

### **2.2.2 Stepper Motor**

For the stepper motor, we successfully control the direction and speed of spinning by using the method we mentioned. However, due to the nature of this motor, the torque was not enough to lift up the window we intended to use at the beginning (a plexiglass). We then used a lighter material, a cardboard, instead of the plexiglass. We also figured that, the more torque you need, the longer time you should set for the for loops in between each pulse sent, however, there is still a saturation point that you can't lift such material if it is too heavy.

The biggest problem we have with the stepper motor and it took us a lot of time to solve is the torque of the motor. We have never expected the motor cannot even lift up a foam board. So, we ended up using a piece of cardboard. One way we can improve the torque of our stepper motor is to implement a very efficient pulley system, since we are doing a project for MPS, so we are not bothering to think about any mechanical ideas.

### **2.2.3 Temperature Sensors**

Using the temperature-sensing units allows us to obtain accurate readings as input for our fuzzy logic program. The op-amp is a crucial part of this unit, since without it, the readings were too low for the A/D converter. Also, it was helpful in that it allowed us to scale the readings into a range

relevant to our purposes. Since we didn't need to distinguish between temperatures beyond 120 degrees or lower than 0 degrees Fahrenheit, while we were using fuzzy logic, we could appropriately limit the range using the op amps characteristics. Our team felt that the temperature sensors were the best choice we have made for this entire project, the unit is easy to use and it's very accurate and efficient.

## **2.2.4 Fuzzy Logic**

The result of the fuzzification procedure produced a smart logical mode for the system to operate in. In this mode which the system will be in most of the time. It will automatically open and close accordingly and turn on and adjust the cooling and heating accordingly. The fuzzification of the system is easier done in C than with assembly. We ran into problems implementing this because of the lack of information we found on this topic. But we were able to research similar products online and come up with our own fuzzy model.

# 3. Back Materials

## 3.1 References

	Document
[1]	“Interfacing a Hitachi HD44780 to a Motorola 68HC12” at the end of the 68HC12 user’s manual
[2]	Software and Hardware Engineering
[3]	Motorola Fast And LS TTL Data

Table 3: References

## 3.2 Bibliography

### 3.2.1 URLs

1. <http://www.cs.uiowa.edu/~jones/step/>
2. [http://www.fdk.co.jp/fdk\\_sale-e/html/shop2-e.html](http://www.fdk.co.jp/fdk_sale-e/html/shop2-e.html)
3. <http://www.newcastle.edu.au/department/av/bilby/stepper.htm>
4. [http://www.parallaxinc.com/html\\_files/component\\_shop/product\\_list.htm](http://www.parallaxinc.com/html_files/component_shop/product_list.htm)
5. [http://www.parallaxinc.com/html\\_files/products/BS\\_Accessories/little\\_step\\_u.asp](http://www.parallaxinc.com/html_files/products/BS_Accessories/little_step_u.asp)
6. <http://www.radioshack.com>
7. <http://www.efunda.com/home.cfm>

### 3.2.2 Printed Materials

All the printed material has been attached at the end of this report.

### 3.3 Appendices

#### 3.3.1 Circuit diagrams

This section includes all the circuit diagrams for our project.

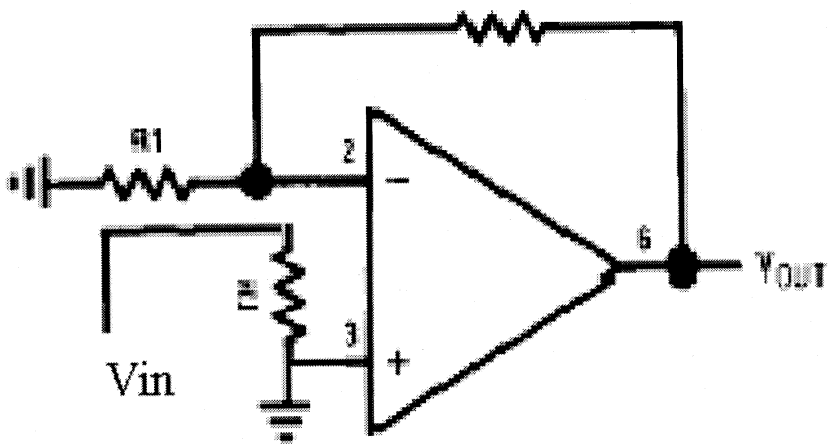
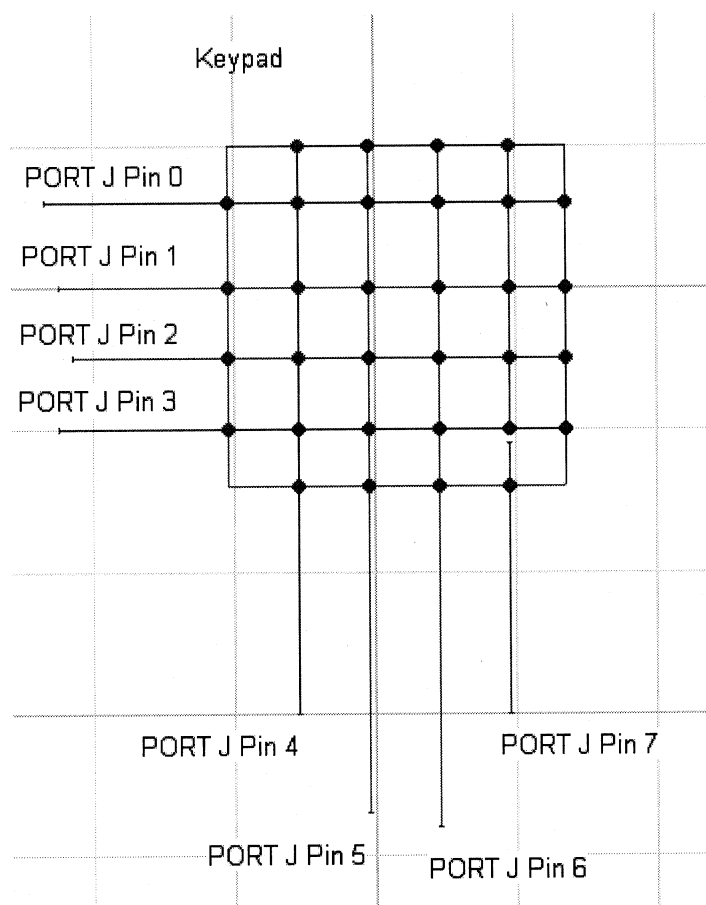
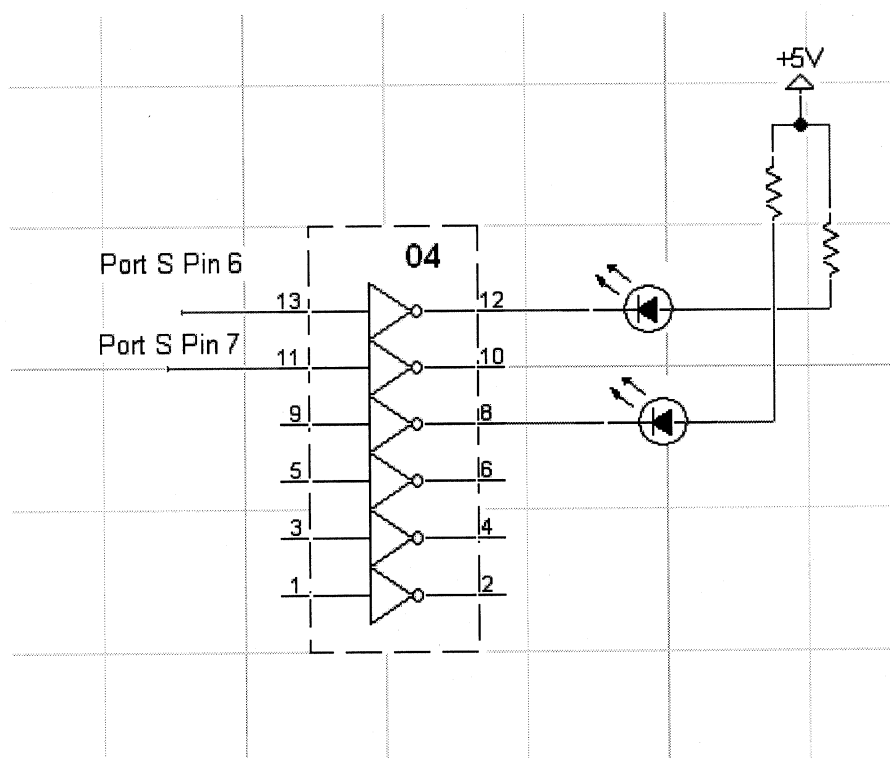


Figure 3: Op-amp circuit for sensor

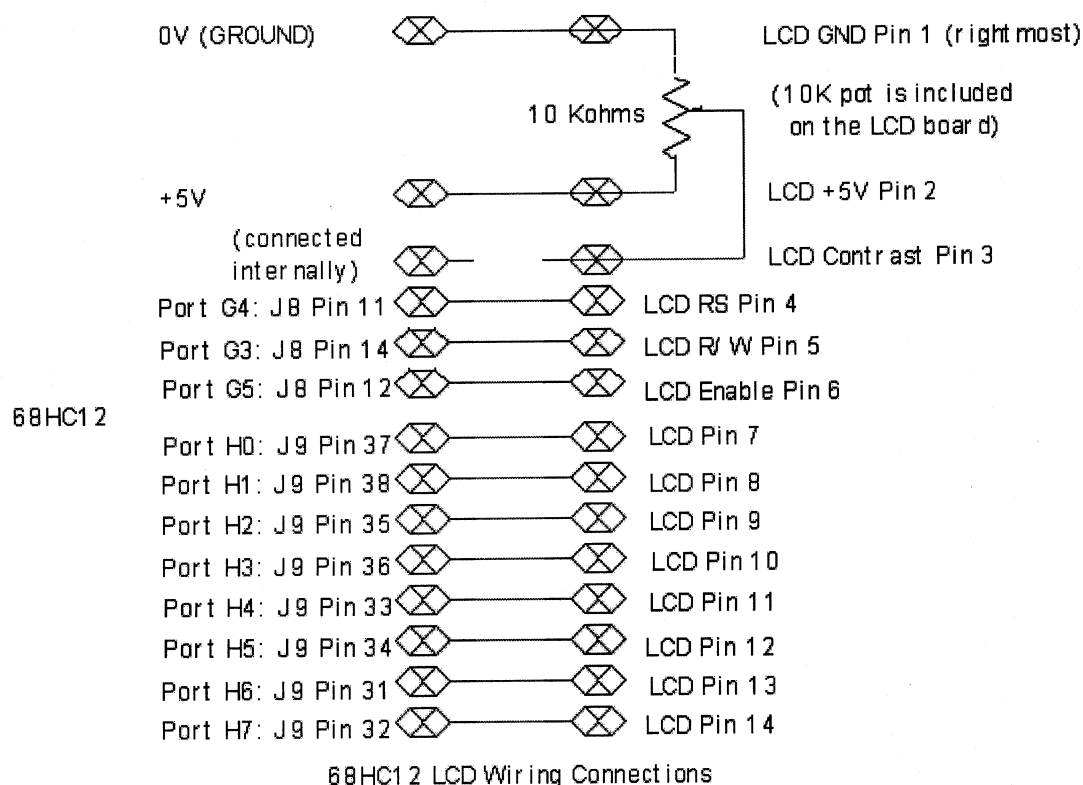


**Figure 4: Keypad Circuit**

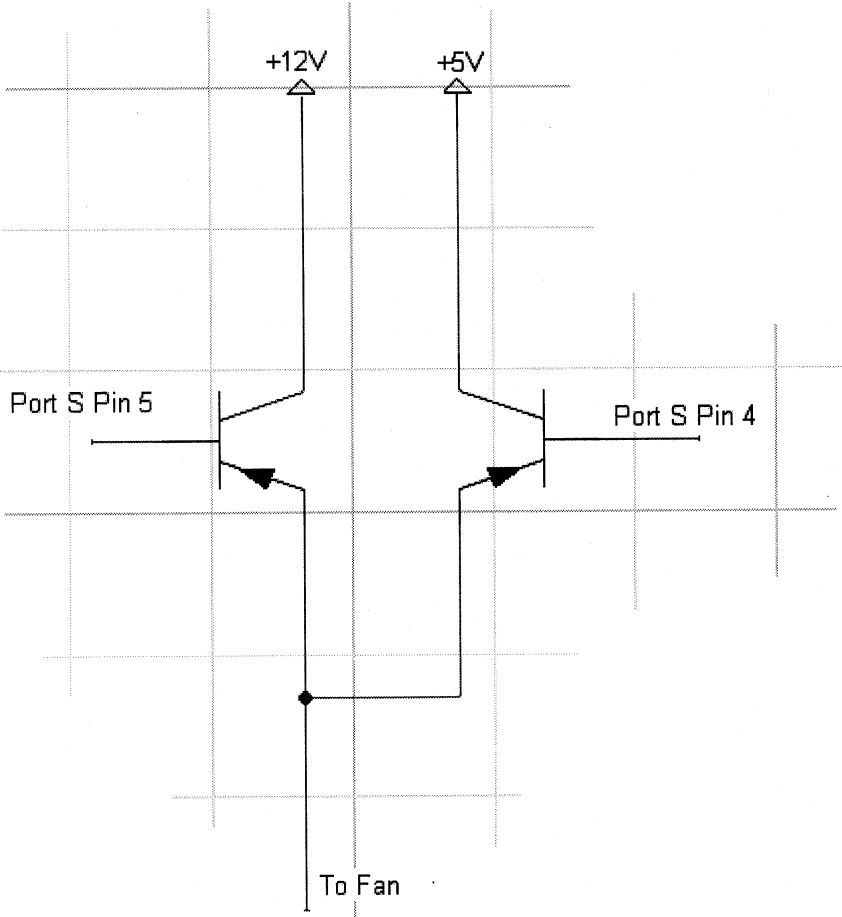




**Figure 5: LEDs Circuit (heat light)**



**Figure 6: LCD Display**



**Figure 7: Fan circuit**

# Stepper Motor Controller

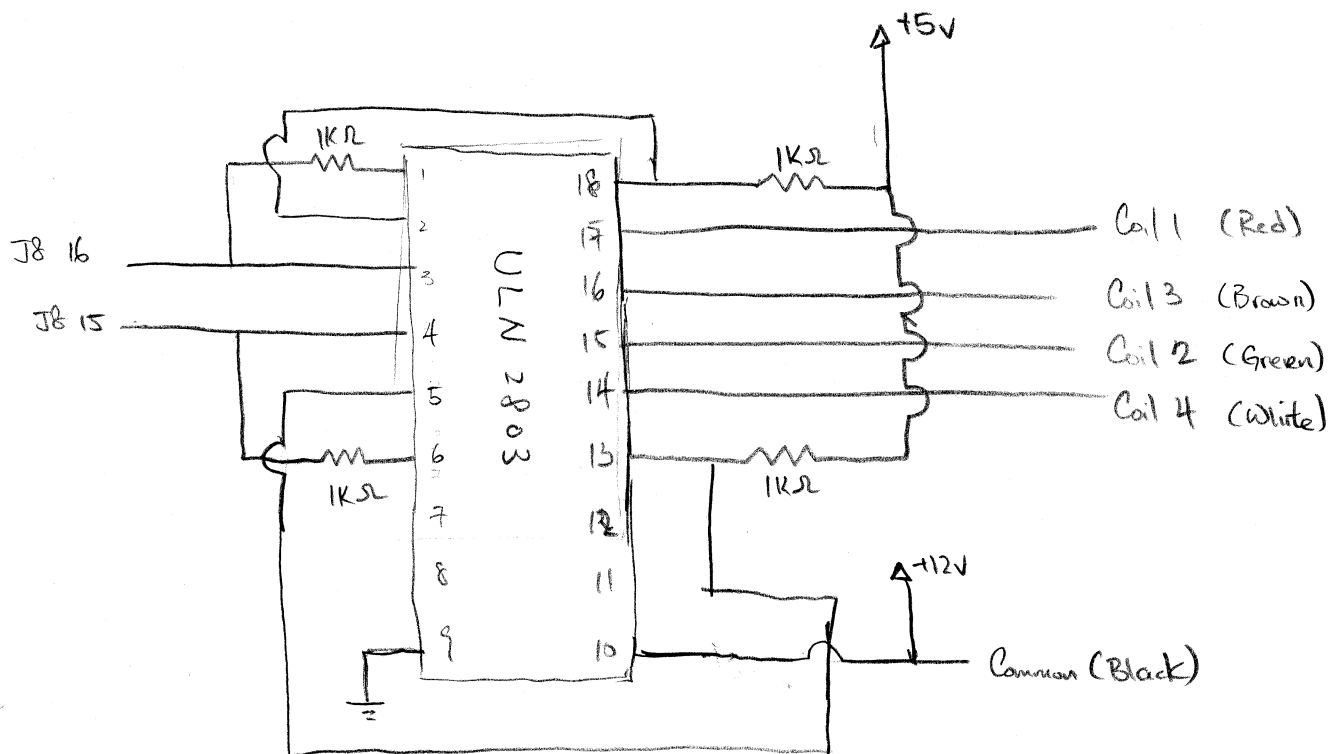


Figure 8.

### 3.3.2 Flow Chart

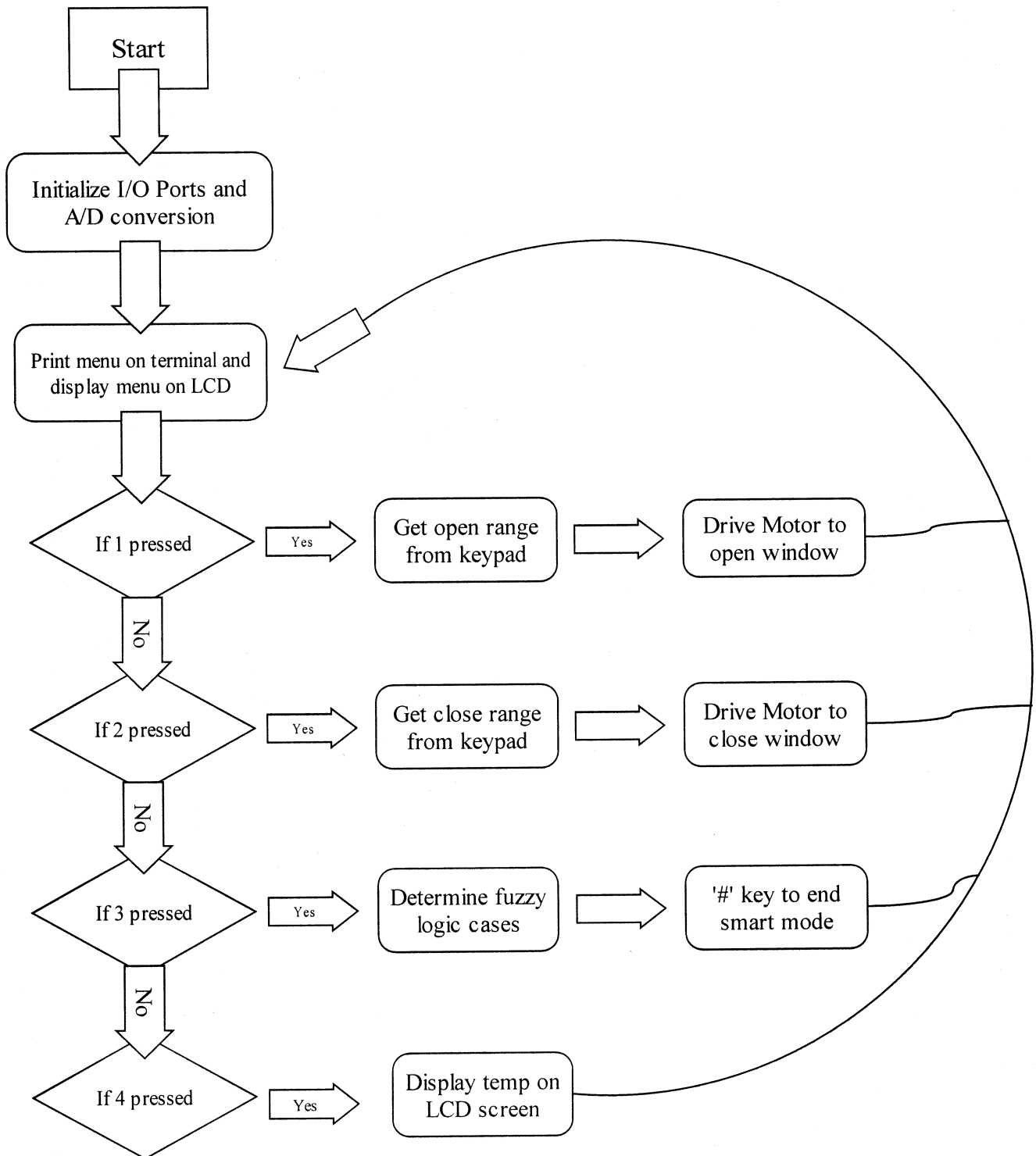


Figure 8: Software Flow Chart

### 3.3.2 Codes

This section includes two source code:

```
#include <hc812a4.h>
#include <introl.h>
#include <debug12.h>
#include <keypad.h>
#include <lcd12.h>
#include <hc912b32.h>

__mod2__ void ADInt();
//__mod2__ void Timer();
int Fuzzify(int temp);
int FL(int tempout, int tempin);
void close_window(int move);
void open_window(int move);
void printMenu(void);
void setUpLCD(void);
void displayMenu(void);
void write2LCD(char string[]);
void randomYesOrNo(void);
void GetTemp(void);
void SmartMode(void);
void OpenMode(void);
void CloseMode(void);
void randomDayOfTheWeek(void);
void randomTrueOrFalse(void);
int intcommand(void);
void writenum2LCD(int num);
void getKeyPadEntry(char string[],int size);

int runtime;
int i;
int j;
int loc;          // 5 is close totally, 0 is open totally
int outside_temp;
int inside_temp;
int smartmode=0;

void __main()
{
    char inString[255];
    /* DB12->SetUserVector(Timer0, Timer);
       _H12INTCR=0xC0;
       _H12TSCR=0x80;
       _H12TMSK2=0x02;
       _H12TIOS=0x01;
       _H12TMSK1=0x01;
       _H12TC0=_H12TCNT+20000;
    */

    DB12->SetUserVector(9,ADInt);
    _H12ADTCTL2 = 0xC2;      //power up ATD
    for ( i =0; i < 1000; i++);      //wait for ATD to power up
```

```

_H12ADTCTL4 = 0x61;
_H12ADTCTL5 = 0x73;    //set MULT =1 for multi-channel operation

loc = 5;
setUpLCD();
InitPortJ();
_H12DDRS = 0xFF;
_H12PORTS = 0x00;

DB12->printf("LCD panel is powering up...    \r\n");
while(1)
{
    DB12->printf("%d\rInside temp: %d        \r\n", inside_temp/7,
inside_temp/7);
    DB12->printf("%d\rOutside temp: %d        \r\n", outside_temp/7,
outside_temp/7);
    runtime = _H12TCNT & 0x00FF;
    runtime = runtime * 113;
    displayMenu();

    getKeyPadEntry(inString,255);
    //DB12->printf("\n\r\n\r\n");
    switch(inString[0])
    {
        case '1':
            OpenMode();
            break;
        case '2':
            CloseMode();
            break;
        case '3':
            SmartMode();
            break;
        case '4':
            GetTemp();
            break;
        default:
            break;
    }
}

}

void setUpLCD(void)
{
    OpenXLCD(0x3f);
    WriteCmdXLCD(0x80);
}

void write2LCD(char string[])
{
    WriteCmdXLCD(0x01);
    WriteBuffer(string);
    for(i=0;i<10;i++)
        for(j=0;j<10000;j++);
}

```

```
void printMenu(void)
{
    char open[] = "1)Open";
    char close[] = "2)Close";
    char smart[] = "3)Smart";
    char tem[] = "4)Temp is";
    WriteCmdXLCD(0x01);

    SetDDRamAddr(0x00);
    WriteBuffer(&open);
    SetDDRamAddr(0x09);
    WriteBuffer(&close);
    SetDDRamAddr(0x40);
    WriteBuffer(&smart);
    SetDDRamAddr(0x49);
    WriteBuffer(&tem);
}

void getKeyPadEntry(char string[],int size)
{
    unsigned int i = 0;
    unsigned char input = '';
    unsigned int done =0;

    while (!done)
    {
        while (!KeyFlag);
        input = Input;
        KeyFlag = 0;
        if(input == '*')
        {
            string[i] = '\\0';
            done = 1;
            break;
        }
        DB12->putchar(input);
        string[i++] = input;
        if(i >= (size -1) )
        {
            string[i] = '\\0';
            done = 1;
            break;
        }
    }
    DB12->putchar('\\n');
    DB12->putchar('\\r');
}

void displayMenu(void)
{
    DB12->printf("-----\\n\\r");
    DB12->printf("1) Open Window\\n\\r");
    DB12->printf("2) Close Window\\n\\r");
    DB12->printf("3) Smart Mode\\n\\r");
}
```

```
    DB12->printf("4) Show current inside Temp\n\r");
    printMenu();
}

void OpenMode(void)
{
    int start = 0;
    int i=0;
    int j=0;
    write2LCD("Open Range?");
    DB12->printf("Open Range?");
    start = intcommand();
    DB12->printf("\n\r");
    if ( start < 0 || start > 5 || loc - start < 0)
    {
        write2LCD("invalid range");
        DB12->printf("invalid range");
    }
    else
    {
        write2LCD("Range: ");
        writenum2LCD(start);
        open_window(start);
    }
    for(i=0;i<100;i++)
        for(j=0;j<5000;j++);
}

void CloseMode(void)
{
    int end = 0;
    int i=0;
    int j = 0;
    write2LCD("Close Range?");
    DB12->printf("Close Range?");
    end = intcommand();
    DB12->printf("\n\r");
    if ( end < 0 || end > 5 || loc+end > 5)
    {
        DB12->printf("invalid range");
        write2LCD("invalid range");
    }
    else
    {
        write2LCD("Range: ");
        writenum2LCD(end);
        close_window(end);
        //DB12->printf("%d\r\nWindow Position is: %d",loc,loc);
    }
    for(i=0;i<100;i++)
        for(j=0;j<5000;j++);
}

void SmartMode(void)
{
    int fuzzytempl;
```



```

int fuzzytemp2;
int i;
int rule;
int movewindow;
char stop= '';
stop = Input;
smartmode =1;
while ( stop != '#')
{
    stop = Input;
    write2LCD("In Smart Mode");
    DB12->printf("\rIn Smart Mode          \r\n");
//
//
    if (counter >= 10)
    {
        if(1)
        {
            //read from AD converter
            _H12ADTCTL5 = 0x53;
            while ( _H12PORTAD & 0x80){}
            _H12ADTCTL2 = 0x80;          //power up ATD
//
//
            while (!( _H12ADTSTAT & 0x80)){}
            inside_temp=1.8* _H12ADR3H;
            outside_temp =1.8* _H12ADR2H;
            DB12->printf("%d\rInside temp: %d          \r\n",
inside_temp/7, inside_temp/7);
            DB12->printf("%d\rOutside temp: %d          \r\n",
outside_temp/7, outside_temp/7);
            DB12->printf("%d\rWindow Position: %d          \r\n",
loc,loc);

            //_H12ADTCTL5 = 0x53;

            fuzzytemp1=Fuzzify(outside_temp);
            fuzzytemp2=Fuzzify(inside_temp);
            //DB12->printf("%d\rDEBUG MESSAGE:
%d\r\n",outside_temp,outside_temp);
            rule = FL(fuzzytemp1, fuzzytemp2);

            switch (rule)
            {
                case 1: _H12PORTS=0x80;          // high
heat , both inside outside cold
                    DB12->printf("In case 1\r\n\r\n");
                    if (loc < 5)
                        close_window(5-loc);

                    break;

                case 2:
                    _H12PORTS=0x40;
                    //low heat , outside cold inside cool
                    DB12->printf("In case 2\r\n\r\n");
                    if (loc < 5)
                        close_window(5-loc);

```

```

break;

case 3:
everything , outside cold inside warm
_H12PORTS=0x00;          // off
DB12->printf("In case 3\r\n\n");
if (loc < 5)
    close_window(5-loc);

break;

case 4:
//low fan , outside cold inside hot
_H12PORTS=0x10;
DB12->printf("In case 4\r\n\n");
if (loc < 5)
    close_window(5-loc);

break;

case 5: _H12PORTS=0x80;    //off everything ,
outside cool inside cold
DB12->printf("In case 5\r\n\n");
if (loc > 0)
    open_window(loc);

break;

case 6: _H12PORTS=0x00;    //off everything ,
outisde cool inside cool
DB12->printf("In case 6\r\n\n");
if ((loc >=0) && (loc<2))
    close_window(2-loc);
if ((loc <=5) && (loc>2))
{
    open_window(loc-2);
}
break;

case 7: _H12PORTS=0x10;    //low fan , outside
cool inside warm || outside hot inside warm
DB12->printf("In case 7\r\n\n");
if ((loc >=0) && (loc<2))
    close_window(2-loc);
if ((loc <=5) && (loc>2))
{
    open_window(loc-2);
}
break;

case 8: _H12PORTS=0x10;    // low fan , outside
cool inside hot
DB12->printf("In case 8\r\n\n");

if ((loc >=0) && (loc<2))
    close_window(3-loc);
if ((loc <=5) && (loc>2))

```

```

                                {
                                    open_window(loc-2);
                                }
                                break;

warm inside cold                case 9: _H12PORTS=0x40;          //low heat, outside
                                DB12->printf("In case 9\r\n\n");
                                if ((loc >=0) && (loc<2))
                                    close_window(3-loc);
                                if ((loc <=5) && (loc>2))
                                    {
                                        open_window(loc-2);
                                    }
                                break;

                                case 10: _H12PORTS=0x00; //off everything,
outside warm inside cool || outside warm inside warm
                                DB12->printf("In case 10\r\n\n");
                                if (loc > 0)
                                    {
                                        open_window(loc);
                                    }
                                break;

inside hot                      case 11: _H12PORTS=0x20; //high fan, outside warm
                                DB12->printf("In case 11\r\n\n");
                                if (loc > 0)
                                    {
                                        open_window(loc);
                                    }
                                break;

                                case 12: _H12PORTS=0x00;          //off
everything, outside hot inside cold //impossible
                                DB12->printf("In case 12\r\n\n");
                                if (loc >0)
                                    {
                                        open_window(loc);
                                    }
                                break;

hot inside warm                case 13: _H12PORTS=0x10;          //low fan, outside
                                DB12->printf("In case 12\r\n\n");
                                if (loc <5 )
                                    {
                                        close_window(5-loc);
                                    }
                                break;

                                case 14: _H12PORTS=0x20;          //high fan,

```

```

outside hot inside hot

                                DB12->printf("In case 12\r\n\n");

                                if (loc <5)
                                {
                                    close_window(5-loc);
                                }
                                break;

                                default:
                                    break;

                                }

                                }

                                }
                                DB12->printf("End of Smart Mode");
                                smartmode =0;

                                }

void GetTemp(void)
{

    int i=0;
    int stupid=0;
    char temp[5];

//        while (_H12PORTAD & 0x80){}
//        for (i=0;i<100;i++);
//        _H12ADTCTL2 = 0x80;        //power up ATD
//        _H12ADTCTL5 = 0x53;
//        while (!( _H12ADTSTAT & 0x80)){}
//        for(i=0;i<100;i++);
//        DB12->printf("%d\rInside Temperature: %d\n\r",inside_temp/7);
//        stupid = inside_temp/7;
//        write2LCD("Temp is: ");
//        writenum2LCD(stupid);
//        WriteBuffer("C");
//        DB12->printf("%d\rInside Temperature is: %d C\n\r",stupid,stupid);
//        _H12ADTCTL5 = 0x53;

    for(i=0;i<100;i++)
        for(j=0;j<5000;j++);

}

int intcommand(void)
{
    char string[255];
    int error = 0;
    int done = 0;
    int i =0;
    int number = 0;
    getKeyPadEntry(string,255);

    while(!done)
    {

```

```
        char c = string[i++];
        if(c == '\\0')
        {
            done =1;
            break;
        }
        if(!isdigit(c))
        {
            done = 1;
            error = 5;
            break;
        }
        number *=10;
        number += c - 48;
    }

    if(error)
    {
        return -1;
    }

    return number;
}

void writenum2LCD(int num)
{
    int i;
    int displayed = 0;
    for(i = 10000;i>0;i/=10)
    {
        int digit = num / i;
        if(digit || displayed)
        {
            WriteDataXLCD(digit + 48);
            displayed = 1;
        }
        num -= i *digit;
    }
}

int Fuzzify(int temp)
{
    if (temp >= 180)
        return 3;
    if ((temp > 134) && (temp < 180))
        return 2;
    if ((temp > 90) && (temp < 135))
        return 1;
    if (temp <= 90)
        return 0;
}

int FL(int tempout, int tempin)
{
    int fuzzrule[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,10,11},{12,7,13,14}};
    return fuzzrule [tempout][tempin];
}
```

```

void open_window(int move)
{
    int j,i;

    _H12DDRG = 0xFF;
    if ( move != 0)
    {
        for ( j=0; j< move*20;j++)
        {
            // clockwise
            _H12PORTG = 0x0F & 0x00;
            for (i=0; i<2000;i++);
            _H12PORTG = 0x0F & 0x02;
            for (i=0; i<2000;i++);
            _H12PORTG = 0x0F & 0x03;
            for (i=0; i<2000;i++);
            _H12PORTG = 0x0F & 0x01;
            for (i=0; i<2000;i++);
        }
    }
    loc = loc - move;
    DB12->printf("%d\rWindow Position: %d\n\r" , loc, loc);
    for(i=0;i<100;i++)
        for(j=0;j<5000;j++);
}

void close_window(int move)
{
    int j,i;

    _H12DDRG = 0xFF;
    if(move != 0)
    {
        for ( j=0; j< move*20;j++)
        {
            // counter-clockwise
            _H12PORTG = 0x0F & 0x00;
            for (i=0; i<2000;i++);
            _H12PORTG = 0x0F & 0x01;
            for (i=0; i<2000;i++);
            _H12PORTG = 0x0F & 0x03;
            for (i=0; i<2000;i++);
            _H12PORTG = 0x0F & 0x02;
            for (i=0; i<2000;i++);
        }
    }
    loc = loc + move;
    DB12->printf("%d\rWindow Position: %d\n\r" , loc, loc);
    for(i=0;i<100;i++)
        for(j=0;j<5000;j++);
}

/*
__mod2__ void Timer()
{
    H12TC0 = H12TC0 + 20000;
}

```

```

    if (smartmode == 1)
        counter++;
    _H12TFLG1=0x01;
    for(i=0;i<100;i++)
        for(j=0;j<5000;j++);
}
*/

__mod2__ void ADInt()
{
    outside_temp = _H12ADR2H;
    inside_temp = _H12ADR3H;
    _H12ADTCTL5 = 0x73;
}

```

Table 4: fuzzywuzzy.c

```

#include <hc812a4.h>
#include <introl.h>
#include <debug12.h>

char Input;
int KeyFlag;
char Pad[4][4] =
{{'D','#','0','*'},{'C','9','8','7'},{'B','6','5','4'},{'A','3','2','1'}};

void InitPortJ(void);
__mod2__ void KeyWakup(void);

void InitPortJ(void){

    DB12->SetUserVector(PortJKey,KeyWakup);
    _H12DDRJ = 0xF0;
    _H12KPOLJ = 0x00;
    _H12KWIFJ = 0xFF;
    _H12PUPSJ = 0xFF;
    _H12PULEJ = 0x0F;
    _H12KWIEJ = 0x0F;
    KeyFlag = 0;
    Input = '';
}

__mod2__ void KeyWakup(void){
    unsigned int column,row;
    unsigned char original;
    unsigned char temp;
    int i;

    KeyFlag = 0;

    original = _H12PORTJ;

    if(!(_H12PORTJ & 0x08)){
        column = 3;
    }
    if(!(_H12PORTJ & 0x04)){

```

```
        column = 2;
    }
    if(!(_H12PORTJ & 0x02)){
        column = 1;
    }
    if(!(_H12PORTJ & 0x01)){
        column = 0;
    }

    _H12PORTJ = (0xF0 | original) & 0x80;
    for (i=0;i<4000;i++);

    if ((_H12PORTJ & 0x0F) == 0x0F){
        row = 3;
    }
    _H12PORTJ = (0xF0 | original) & 0x40;
    for (i=0;i<4000;i++);

    if ((_H12PORTJ & 0x0F) == 0x0F){
        row = 2;
    }
    _H12PORTJ = (0xF0 | original) & 0x20;
    for (i=0;i<4000;i++);

    if ((_H12PORTJ & 0x0F) == 0x0F){
        row = 1;
    }
    _H12PORTJ = (0xF0 | original) & 0x10;
    for (i=0;i<4000;i++);

    if ((_H12PORTJ & 0x0F) == 0x0F){
        row = 0;
    }
    Input = Pad[row][column];
    KeyFlag = 1;

    _H12PORTJ = 0x00;
    _H12KWIFJ = _H12KWIFJ;
}
```

*Table 5: Keypad.h*



# Conclusion

Our team had successfully accomplished our tasks stated in the project proposal. After finishing the project, we gain knowledge in controlling the MC68HC12, along with our usage of temperature sensor, stepper motor and fuzzy logic. We also learned to manage our time as the project goes along. Doing researches and reading tutorials from difference references became one of our main tasks in this project, which we benefit from this and gain experience as a real engineer. And we can see our project with the potential and expandability of this project, we see that our WIN2K1 will give users convenience and upgrade their standard of living. Last but not least, we have to make a special thanks to Prof. Kraft, Tong , and Shivani during the whole semester, we have received sufficient and inspiring information.

## Lab 6 Temperature Sensor

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, published by Brooks-Cole, copyright © 2000.

- Goals**
- Design the hardware interface between a DS1620 temperature sensor and a microcomputer,
  - Implement synchronous serial communication using simple I/O directly to the clock and data pins,
  - Create the low-level device driver that could be used in other applications.

- Review**
- Valvano Section 3.3 about gadfly synchronization,
  - Valvano Section 3.4.2 about accurate time delays,
  - Valvano Section 3.4.8 about handshaking with the DS1620,
  - Reread Lab 1 about binary fixed point format,
  - DS1620 data sheets included with this Lab Manual,
  - The chapter on the parallel port and output compare in the Motorola Reference Manual.
- Starter files**
- DS1620.C, DS1620.H, DSTEST.C

### Background

One of the basic building components of a microprocessor-based control system is the sensor. In this lab, you will interface a DS1620 to your computer, and use it as part of a temperature controller. We will simulate a digital control system that applies heat to the room in order to maintain the temperature as close to a desired temperature setpoint,  $T^*$ , as possible. This is a closed loop control system because the control signals (heat) depend on the state variables (temperature). Your system will communicate with the DS1620 to estimate the current temperature,  $T'$ . In this application, the actuator has only two states: *on* that warms up the room and *off* that does not apply heat. Read about the operation of the DS1620 in general and the  $T_{COM}$  signal in particular. For this control problem to function properly there must be a passive heat loss that lowers the room temperature when the heater is turned off. A typical digital control algorithm for this type of actuator is Bang-Bang. Other names for Bang-Bang include Two-position, On-off, or Binary Controller. There are two setpoint temperatures in a Bang-Bang controller,  $T_{HIGH}$  and  $T_{LOW}$ . The controller turns on the power (activate relay) if the temperature goes below  $T_{LOW}$  and turns off the power (deactivate relay) if the temperature goes above  $T_{HIGH}$ . The difference  $T_{HIGH} - T_{LOW}$  is called hysteresis. Hysteresis extends the life of the relay by reducing the number of times the relay opens and closes.

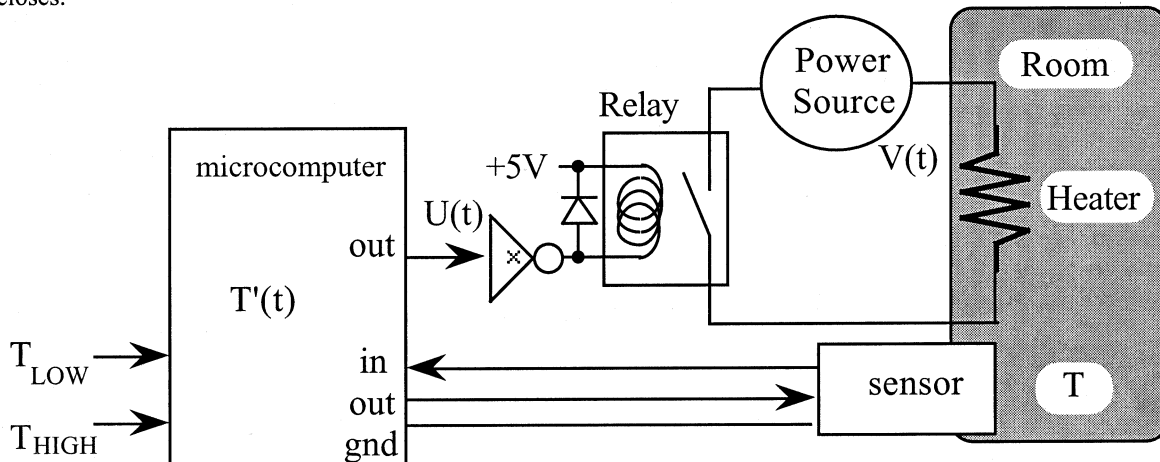


Figure 6.1. General Microcomputer-based Temperature Controller

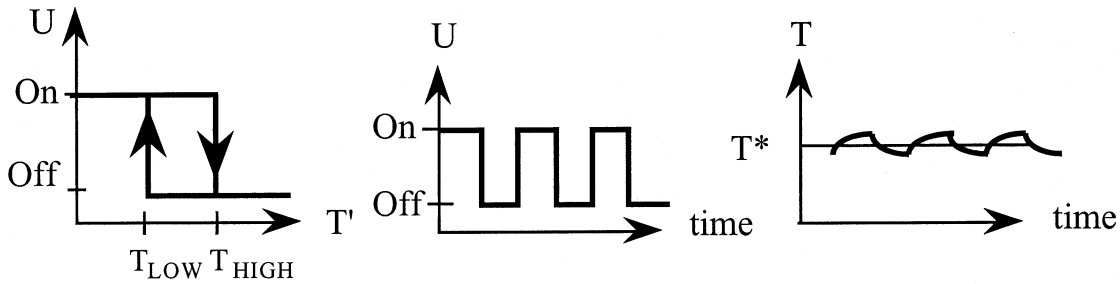


Figure 6.2. Algorithm for Bang-Bang Temperature Controller

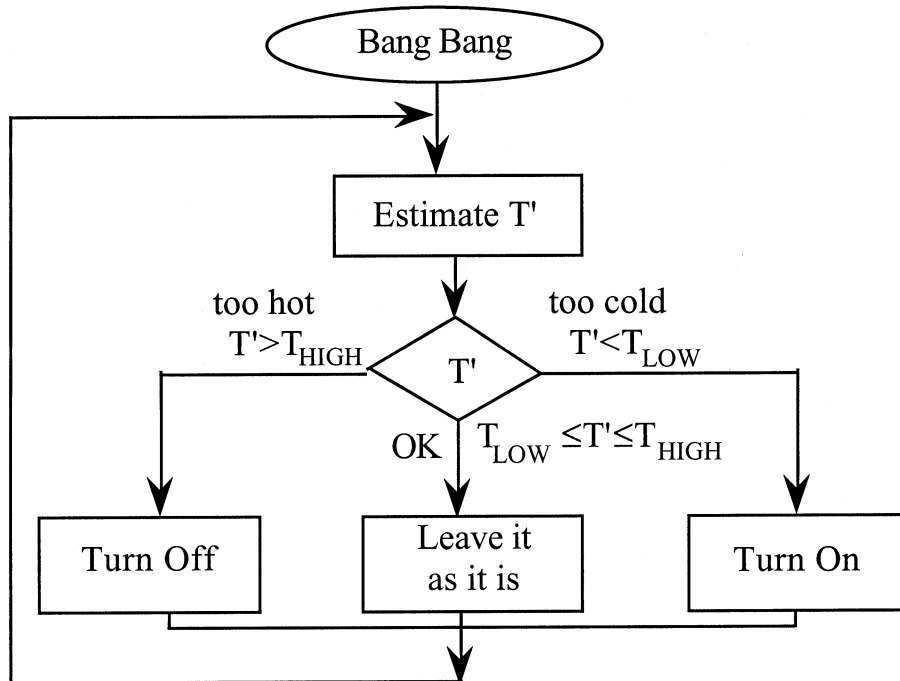


Figure 6.3. Software Algorithm for Software Based Bang-Bang Temperature Controller

Once programmed with the two setpoint temperatures,  $T_{HIGH}$  and  $T_{LOW}$ , the DS1620 will perform the above bang-bang algorithm automatically. The following figure shows an actual DS1620-based controller. The DS1620 can be programmed at the factory before installing the chip into a system. Here it is shown with a microcomputer that allows the operator to adjust the setpoints.

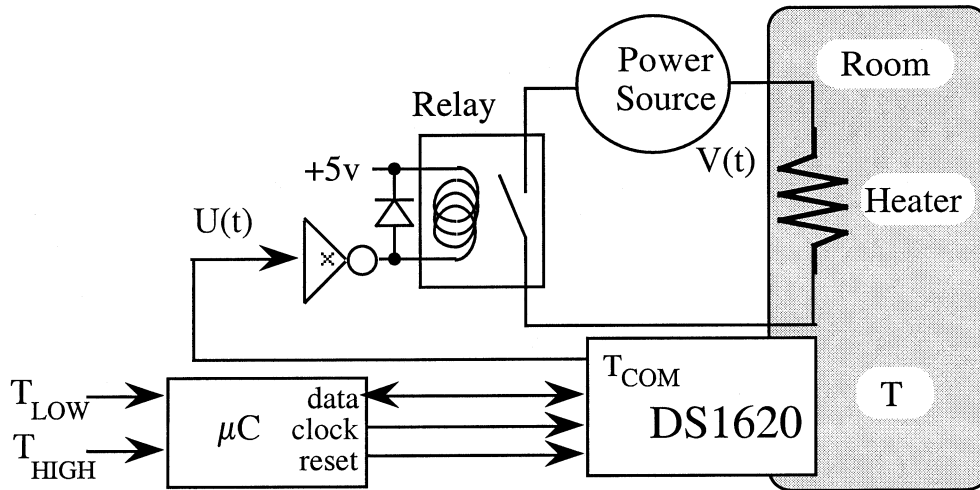


Figure 6.4. DS1620-based Temperature Controller

Instead of a relay and heater, you will connect three LED's to the DS1620. The middle LED simulates the control to the heater that would add or not add thermal energy to the room. The other two will help debug your system. The two setpoint temperatures,  $T_{HIGH}$  and  $T_{LOW}$ , will be entered using the `InFDec()` routine you developed in Lab 1. The microcomputer will send the two setpoints to the DS1620. Continuous mode will be started, and you should be able to observe the controller action on the three LED's. There are five types of communications that you can perform with the DS1610. See Table 3 and Figures 3,4 of the DS1620 data sheets. Information is sent LSB first.

#### 1) Execute Function

This type of communication involves sending an 8-bit instruction from the 6812 to the DS1620 and no data. The two examples of this type are `StartConvertT (0xEE)` and `StopConvertT (0x22)`. For these commands, you simply send the 8 bits.

#### 2) Send Command

This type of communication involves sending both an 8-bit instruction and an 8-bit command from the 6812 to the DS1620. The only example of this type is `WriteConfig`. For this command, you first send the 8 bits (0x0C), then you send the 8 bits of data.

#### 3) Receive Status

This type of communication involves first sending an 8-bit instruction to the DS1620 then receiving back an 8 bit data from the DS1620. The only example of this type is `ReadConfig`. For this command, you first send the 8 bits, next you switch the direction register bit for the data pin so it is an input, and then you receive the 9 bits of data.

#### 4) Send Data

This type of communication involves sending both an 8-bit instruction and a 9-bit data from the 6812 to the DS1620. The two examples of this type are `WriteTH` and `WriteTL`. For these commands, you first send the 8 bits, then you send the 9 bits of data.

#### 5) Receive Data

This type of communication involves first sending an 8-bit instruction to the DS1620 then receiving back a 9 bit data from the DS1620. The three examples of this type are `ReadTH`, `ReadTL` and `ReadTemperature`. For these commands, you first send the 8 bits, next you switch the direction register bit for the data pin so it is an input, and then you receive the 9 bits of data.

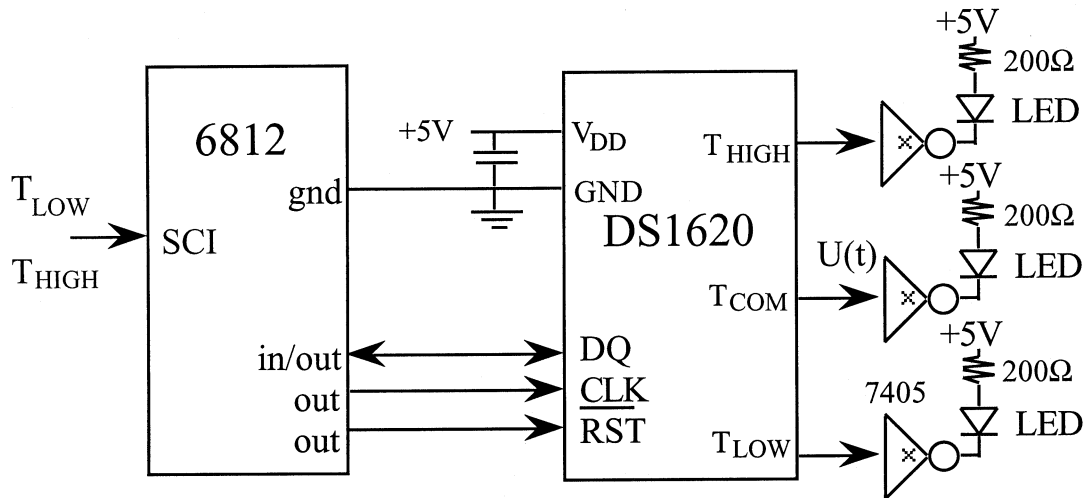


Figure 6.5. Simulated microcomputer-based temperature controller that you will build.

### Preparation

Show the required hardware connections. Label all hardware chips, pin numbers, and resistor values. Ask your TA for the name and location of a demo program that communicates with the DS1620. Modify the port and bit locations to make your hardware.

Write the low-level DS1620 software interface routines. At the lowest level you should be able to send commands and send/receive data. At the next level you will develop commands to read temperature and write setpoints. Pass data into/out of these programs using signed 16-bit binary fixed-point format. Refer to Table 1 of the DS1620 data sheets and Table 1.1 in Lab1 for more information about this binary fixed-point format. You must have a separate DS1620.H and DS1620.C files to simplify the reuse of these routines. You are not allowed to perform serial port I/O (e.g., `InFDec OutString printf`) within the DS1620.C files. These operator interactions will occur in the main program. Write a main program that inputs desired temperature setpoints from the user, and transmits them to the DS1620. Implement a simple interpreter that allows the operator to perform each of the individual 9 operations with the DS1610. In addition, add a thermometer mode that runs a continuous loop repeating these steps over and over until the operator stops the command (use `InStat us`)

- read the current temperature from the DS1620,
- convert the temperature to °F and displays it on the PC screen (using `OutFDec`).

### Procedure

Run the demo program to test the hardware interface. Start with the lowest level routines and test your DS1620.C functions in small pieces. Write a main program that performs the same operation over and over so that you can observe the synchronous serial communication on a dual channel scope.

### Checkout

You should be able to demonstrate your ability to execute all 9 functions individually. Connect the Dual Channel scope to CLK,DQ and explain the signals generated when running thermometer mode.

### Hints

- 1) Make sure the wires are securely attached to your board.
- 2) You can increase the temperature of the DS1620 with your finger, and decrease it with a fan. You could use one of those frozen cubes you put in your cooler, but I suggest you avoid using liquids (e.g., ice) in this lab.

**MOTOROLA**

# Octal High Voltage, High Current Darlington Transistor Arrays

The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open-collector outputs and free wheeling clamp diodes for transient suppression.

The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 6 to 15 volt high level CMOS or PMOS.

**MAXIMUM RATINGS** ( $T_A = 25^\circ\text{C}$  and rating apply to any one device in the package, unless otherwise noted.)

Rating	Symbol	Value	Unit
Output Voltage	$V_O$	50	V
Input Voltage (Except ULN2801)	$V_I$	30	V
Collector Current – Continuous	$I_C$	500	mA
Base Current – Continuous	$I_B$	25	mA
Operating Ambient Temperature Range	$T_A$	0 to +70	$^\circ\text{C}$
Storage Temperature Range	$T_{stg}$	-55 to +150	$^\circ\text{C}$
Junction Temperature	$T_J$	125	$^\circ\text{C}$

$R_{\theta JA} = 55^\circ\text{C/W}$

Do not exceed maximum current limit per driver.

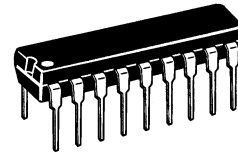
## ORDERING INFORMATION

Device	Characteristics		
	Input Compatibility	$V_{CE}(\text{Max})/I_C(\text{Max})$	Operating Temperature Range
ULN2803A	TTL, 5.0 V CMOS	50 V/500 mA	$T_A = 0 \text{ to } +70^\circ\text{C}$
ULN2804A	6 to 15 V CMOS, PMOS		

# ULN2803 ULN2804

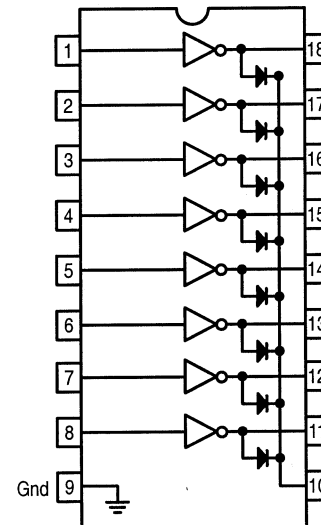
## OCTAL PERIPHERAL DRIVER ARRAYS

### SEMICONDUCTOR TECHNICAL DATA



**A SUFFIX**  
PLASTIC PACKAGE  
CASE 707

## PIN CONNECTIONS



# ULN2803 ULN2804

## ELECTRICAL CHARACTERISTICS ( $T_A = 25^\circ\text{C}$ , unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Output Leakage Current (Figure 1) ( $V_O = 50\text{ V}$ , $T_A = +70^\circ\text{C}$ ) ( $V_O = 50\text{ V}$ , $T_A = +25^\circ\text{C}$ ) ( $V_O = 50\text{ V}$ , $T_A = +70^\circ\text{C}$ , $V_I = 6.0\text{ V}$ ) ( $V_O = 50\text{ V}$ , $T_A = +70^\circ\text{C}$ , $V_I = 1.0\text{ V}$ )	All Types All Types ULN2802 ULN2804	$I_{CEX}$	— — — —	— — — —	100 50 500 500	$\mu\text{A}$
Collector-Emitter Saturation Voltage (Figure 2) ( $I_C = 350\text{ mA}$ , $I_B = 500\text{ }\mu\text{A}$ ) ( $I_C = 200\text{ mA}$ , $I_B = 350\text{ }\mu\text{A}$ ) ( $I_C = 100\text{ mA}$ , $I_B = 250\text{ }\mu\text{A}$ )	All Types All Types All Types	$V_{CE(sat)}$	— — —	1.1 0.95 0.85	1.6 1.3 1.1	V
Input Current – On Condition (Figure 4) ( $V_I = 17\text{ V}$ ) ( $V_I = 3.85\text{ V}$ ) ( $V_I = 5.0\text{ V}$ ) ( $V_I = 12\text{ V}$ )	ULN2802 ULN2803 ULN2804 ULN2804	$I_{I(on)}$	— — — —	0.82 0.93 0.35 1.0	1.25 1.35 0.5 1.45	mA
Input Voltage – On Condition (Figure 5) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 300\text{ mA}$ ) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 200\text{ mA}$ ) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 250\text{ mA}$ ) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 300\text{ mA}$ ) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 125\text{ mA}$ ) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 200\text{ mA}$ ) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 275\text{ mA}$ ) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 350\text{ mA}$ )	ULN2802 ULN2803 ULN2803 ULN2803 ULN2804 ULN2804 ULN2804 ULN2804	$V_{I(on)}$	— — — — — — — —	— — — — — — — —	13 2.4 2.7 3.0 5.0 6.0 7.0 8.0	V
Input Current – Off Condition (Figure 3) ( $I_C = 500\text{ }\mu\text{A}$ , $T_A = +70^\circ\text{C}$ )	All Types	$I_{I(off)}$	50	100	—	$\mu\text{A}$
DC Current Gain (Figure 2) ( $V_{CE} = 2.0\text{ V}$ , $I_C = 350\text{ mA}$ )	ULN2801	$h_{FE}$	1000	—	—	—
Input Capacitance		$C_I$	—	15	25	pF
Turn-On Delay Time (50% $E_I$ to 50% $E_O$ )		$t_{on}$	—	0.25	1.0	$\mu\text{s}$
Turn-Off Delay Time (50% $E_I$ to 50% $E_O$ )		$t_{off}$	—	0.25	1.0	$\mu\text{s}$
Clamp Diode Leakage Current (Figure 6) ( $V_R = 50\text{ V}$ )	$T_A = +25^\circ\text{C}$ $T_A = +70^\circ\text{C}$	$I_R$	—	—	50 100	$\mu\text{A}$
Clamp Diode Forward Voltage (Figure 7) ( $I_F = 350\text{ mA}$ )		$V_F$	—	1.5	2.0	V

# ULN2803 ULN2804

## TEST FIGURES

(See Figure Numbers in Electrical Characteristics Table)

Figure 1.

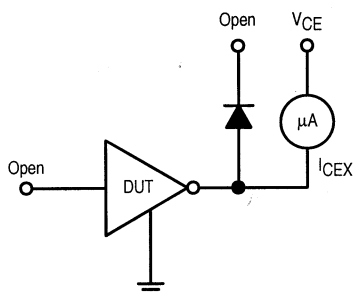


Figure 2.

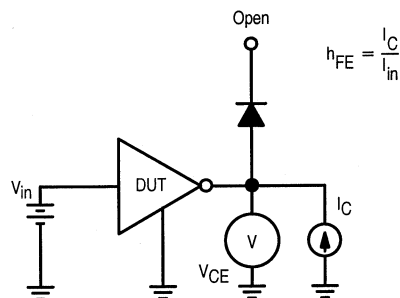


Figure 3.

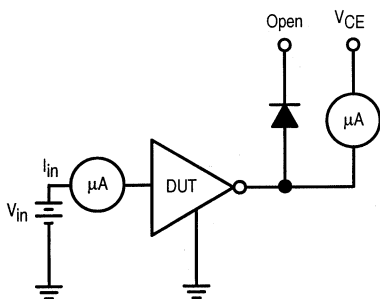


Figure 4.

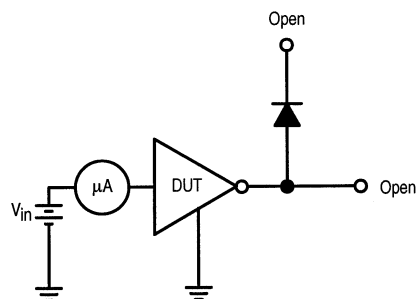


Figure 5.

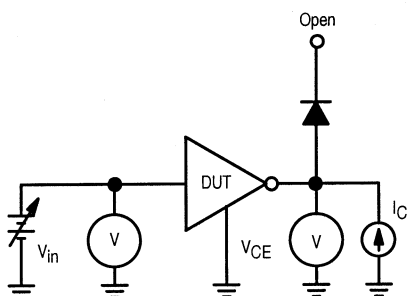


Figure 6.

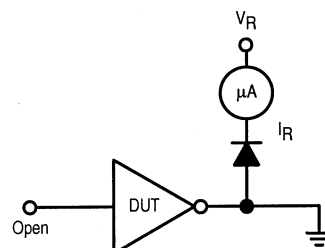
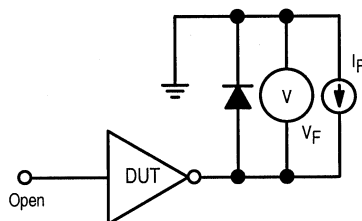


Figure 7.





# ULN2803 ULN2804

TYPICAL CHARACTERISTIC CURVES –  $T_A = 25^\circ\text{C}$ , unless otherwise noted  
Output Characteristics

Figure 8. Output Current versus Saturation Voltage

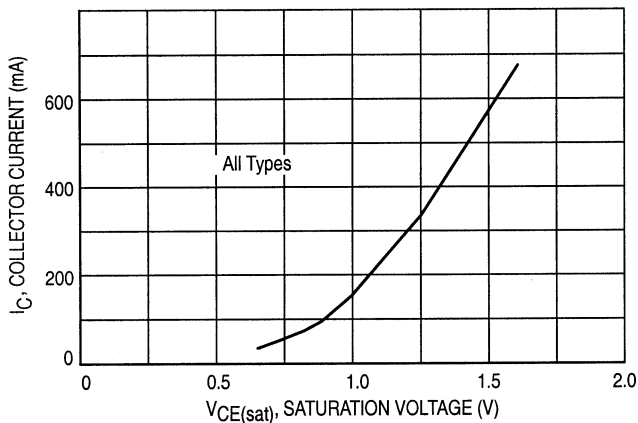
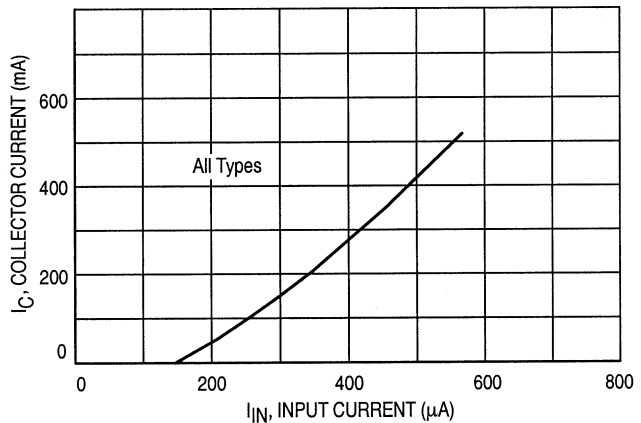


Figure 9. Output Current versus Input Current



## Input Characteristics

Figure 10. ULN2803 Input Current versus Input Voltage

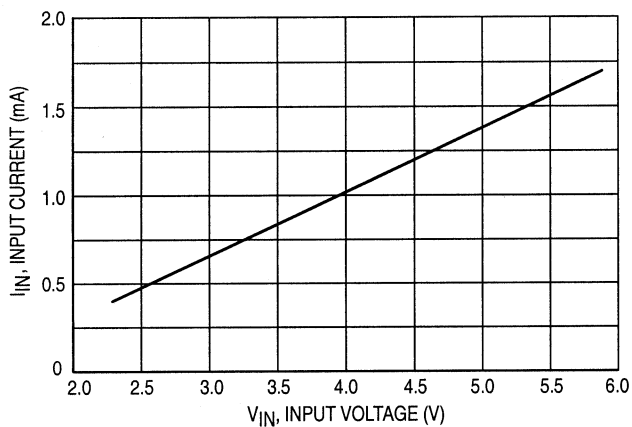


Figure 11. ULN2804 Input Current versus Input Voltage

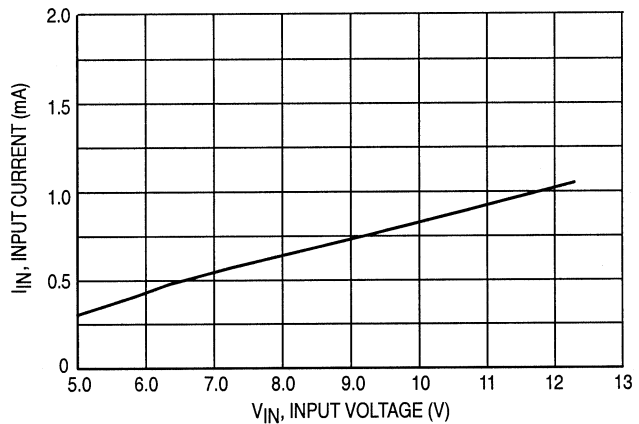
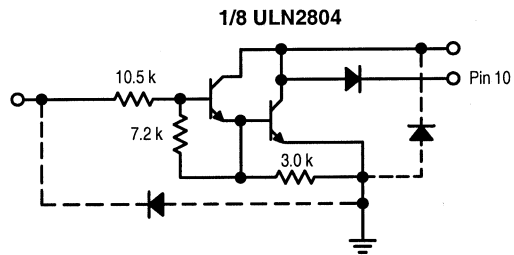
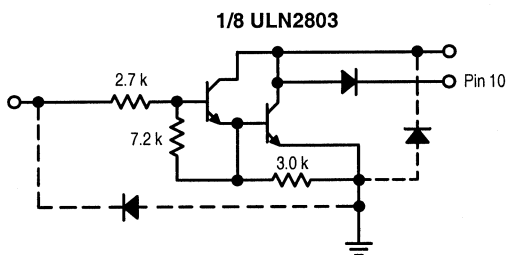
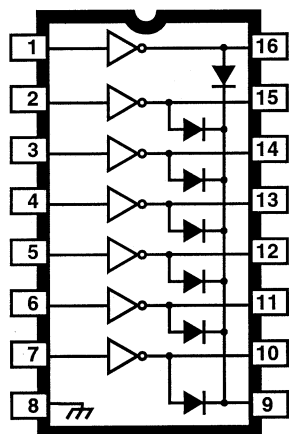


Figure 12. Representative Schematic Diagrams



## HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON ARRAYS



Dwg. No. A-9594

Note that the ULN20xxA series (dual in-line package) and ULN20xxL series (small-outline IC package) are electrically identical and share a common terminal number assignment.

### ABSOLUTE MAXIMUM RATINGS

Output Voltage, $V_{CE}$	
(ULN200xA and ULN200xL) .....	50 V
(ULN202xA and ULN202xL) .....	95 V
Input Voltage, $V_{IN}$ .....	30 V
Continuous Output Current,	
$I_C$ .....	500 mA
Continuous Input Current, $I_{IN}$ .....	25 mA
Power Dissipation, $P_D$	
(one Darlington pair) .....	1.0 W
(total package) .....	See Graph
Operating Temperature Range,	
$T_A$ .....	-20°C to +85°C
Storage Temperature Range,	
$T_S$ .....	-55°C to +150°C

Ideally suited for interfacing between low-level logic circuitry and multiple peripheral power loads, the Series ULN20xxA/L high-voltage, high-current Darlington arrays feature continuous load current ratings to 500 mA for each of the seven drivers. At an appropriate duty cycle depending on ambient temperature and number of drivers turned ON simultaneously, typical power loads totaling over 230 W (350 mA x 7, 95 V) can be controlled. Typical loads include relays, solenoids, stepping motors, magnetic print hammers, multiplexed LED and incandescent displays, and heaters. All devices feature open-collector outputs with integral clamp diodes.

The ULN2003A/L and ULN2023A/L have series input resistors selected for operation directly with 5 V TTL or CMOS. These devices will handle numerous interface needs — particularly those beyond the capabilities of standard logic buffers.

The ULN2004A/L and ULN2024A/L have series input resistors for operation directly from 6 to 15 V CMOS or PMOS logic outputs.

The ULN2003A/L and ULN2004A/L are the standard Darlington arrays. The outputs are capable of sinking 500 mA and will withstand at least 50 V in the OFF state. Outputs may be paralleled for higher load current capability. The ULN2023A/L and ULN2024A/L will withstand 95 V in the OFF state.

These Darlington arrays are furnished in 16-pin dual in-line plastic packages (suffix "A") and 16-lead surface-mountable SOICs (suffix "L"). All devices are pinned with outputs opposite inputs to facilitate ease of circuit board layout. All devices are rated for operation over the temperature range of -20°C to +85°C. Most (see matrix, next page) are also available for operation to -40°C; to order, change the prefix from "ULN" to "ULQ".

### FEATURES

- TTL, DTL, PMOS, or CMOS-Compatible Inputs
- Output Current to 500 mA
- Output Voltage to 95 V
- Transient-Protected Outputs
- Dual In-Line Plastic Package or Small-Outline IC Package

x = digit to identify specific device. Characteristic shown applies to family of devices with remaining digits as shown. See matrix on next page.

# 2003 THRU 2024 HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON ARRAYS

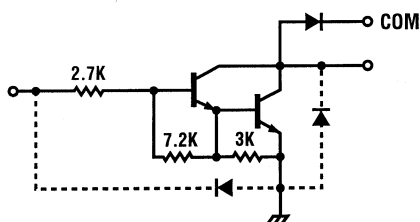
## DEVICE PART NUMBER DESIGNATION

$V_{CE(MAX)}$	50 V	95 V
$I_{C(MAX)}$	500 mA	500 mA
Logic	Part Number	
5V TTL, CMOS	ULN2003A* ULN2003L*	ULN2023A* ULN2023L
6-15 V CMOS, PMOS	ULN2004A* ULN2004L*	ULN2024A ULN2024L

\* Also available for operation between -40°C and +85°C. To order, change prefix from "ULN" to "ULQ".

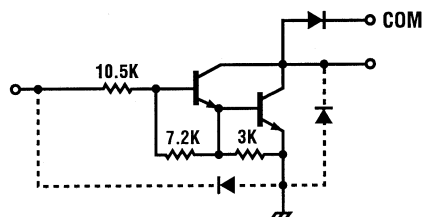
## PARTIAL SCHEMATICS

### ULN20x3A/L (Each Driver)

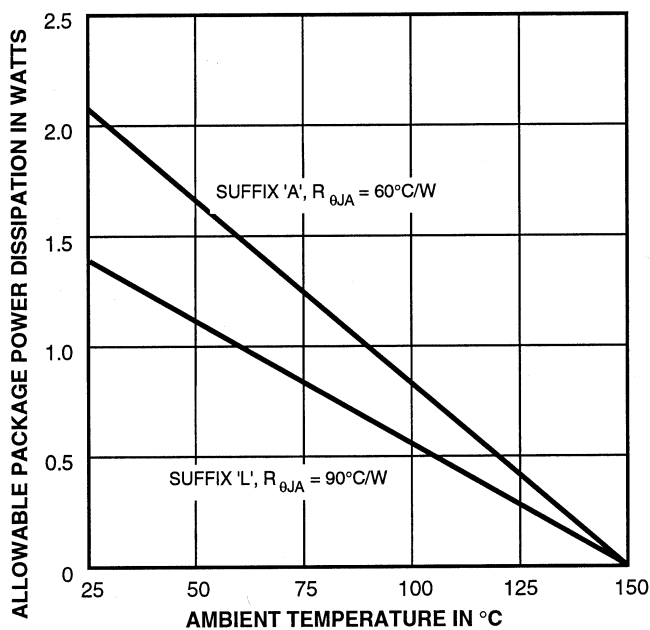


Dwg. No. A-9651

### ULN20x4A/L (Each Driver)



Dwg. No. A-9898A



Dwg. GP-006A

X = Digit to identify specific device. Specification shown applies to family of devices with remaining digits as shown. See matrix above.



115 Northeast Cutoff, Box 15036  
Worcester, Massachusetts 01615-0036 (508) 853-5000  
Copyright © 1974, 1998 Allegro MicroSystems, Inc.

**2003 THRU 2024**  
**HIGH-VOLTAGE,**  
**HIGH-CURRENT**  
**DARLINGTON ARRAYS**

**Types ULN2003A, ULN2003L, ULN2004A, and ULN2004L**

**ELECTRICAL CHARACTERISTICS at +25°C (unless otherwise noted).**

Characteristic	Symbol	Test Fig.	Applicable Devices	Test Conditions	Limits			
					Min.	Typ.	Max.	Units
Output Leakage Current	$I_{CEX}$	1A	All	$V_{CE} = 50\text{ V}, T_A = 25^\circ\text{C}$	—	< 1	50	$\mu\text{A}$
				$V_{CE} = 50\text{ V}, T_A = 70^\circ\text{C}$	—	< 1	100	$\mu\text{A}$
		1B	ULN2004A/L	$V_{CE} = 50\text{ V}, T_A = 70^\circ\text{C}, V_{IN} = 1.0\text{ V}$	—	< 5	500	$\mu\text{A}$
Collector-Emitter Saturation Voltage	$V_{CE(SAT)}$	2	All	$I_C = 100\text{ mA}, I_B = 250\text{ }\mu\text{A}$	—	0.9	1.1	V
				$I_C = 200\text{ mA}, I_B = 350\text{ }\mu\text{A}$	—	1.1	1.3	V
				$I_C = 350\text{ mA}, I_B = 500\text{ }\mu\text{A}$	—	1.3	1.6	V
Input Current	$I_{IN(ON)}$	3	ULN2003A/L	$V_{IN} = 3.85\text{ V}$	—	0.93	1.35	mA
			ULN2004A/L	$V_{IN} = 5.0\text{ V}$	—	0.35	0.5	mA
				$V_{IN} = 12\text{ V}$	—	1.0	1.45	mA
	$I_{IN(OFF)}$	4	All	$I_C = 500\text{ }\mu\text{A}, T_A = 70^\circ\text{C}$	50	65	—	$\mu\text{A}$
Input Voltage	$V_{IN(ON)}$	5	ULN2003A/L	$V_{CE} = 2.0\text{ V}, I_C = 200\text{ mA}$	—	—	2.4	V
				$V_{CE} = 2.0\text{ V}, I_C = 250\text{ mA}$	—	—	2.7	V
				$V_{CE} = 2.0\text{ V}, I_C = 300\text{ mA}$	—	—	3.0	V
		5	ULN2004A/L	$V_{CE} = 2.0\text{ V}, I_C = 125\text{ mA}$	—	—	5.0	V
				$V_{CE} = 2.0\text{ V}, I_C = 200\text{ mA}$	—	—	6.0	V
				$V_{CE} = 2.0\text{ V}, I_C = 275\text{ mA}$	—	—	7.0	V
				$V_{CE} = 2.0\text{ V}, I_C = 350\text{ mA}$	—	—	8.0	V
Input Capacitance	$C_{IN}$	—	All		—	15	25	pF
Turn-On Delay	$t_{PLH}$	8	All	$0.5 E_{IN}$ to $0.5 E_{OUT}$	—	0.25	1.0	$\mu\text{s}$
Turn-Off Delay	$t_{PHL}$	8	All	$0.5 E_{IN}$ to $0.5 E_{OUT}$	—	0.25	1.0	$\mu\text{s}$
Clamp Diode Leakage Current	$I_R$	6	All	$V_R = 50\text{ V}, T_A = 25^\circ\text{C}$	—	—	50	$\mu\text{A}$
				$V_R = 50\text{ V}, T_A = 70^\circ\text{C}$	—	—	100	$\mu\text{A}$
Clamp Diode Forward Voltage	$V_F$	7	All	$I_F = 350\text{ mA}$	—	1.7	2.0	V

Complete part number includes suffix to identify package style: A = DIP, L = SOIC.

**2003 THRU 2024**  
**HIGH-VOLTAGE,**  
**HIGH-CURRENT**  
**DARLINGTON ARRAYS**

**Types ULN2023A, ULN2023L, ULN2024A, and ULN2024L**  
**ELECTRICAL CHARACTERISTICS at +25°C (unless otherwise noted).**

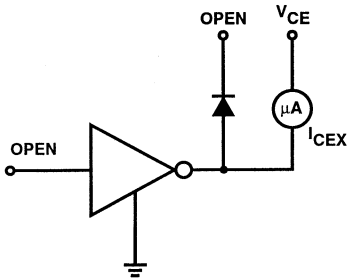
Characteristic	Symbol	Test Fig.	Applicable Devices	Test Conditions	Limits			
					Min.	Typ.	Max.	Units
Output Leakage Current	$I_{CEX}$	1A	All	$V_{CE} = 95\text{ V}, T_A = 25^\circ\text{C}$	—	< 1	50	$\mu\text{A}$
				$V_{CE} = 95\text{ V}, T_A = 70^\circ\text{C}$	—	< 1	100	$\mu\text{A}$
		1B	ULN2024A/L	$V_{CE} = 95\text{ V}, T_A = 70^\circ\text{C}, V_{IN} = 1.0\text{ V}$	—	< 5	500	$\mu\text{A}$
Collector-Emitter Saturation Voltage	$V_{CE(SAT)}$	2	All	$I_C = 100\text{ mA}, I_B = 250\text{ }\mu\text{A}$	—	0.9	1.1	V
				$I_C = 200\text{ mA}, I_B = 350\text{ }\mu\text{A}$	—	1.1	1.3	V
				$I_C = 350\text{ mA}, I_B = 500\text{ }\mu\text{A}$	—	1.3	1.6	V
Input Current	$I_{IN(ON)}$	3	ULN2023A/L	$V_{IN} = 3.85\text{ V}$	—	0.93	1.35	mA
			ULN2024A/L	$V_{IN} = 5.0\text{ V}$	—	0.35	0.5	mA
				$V_{IN} = 12\text{ V}$	—	1.0	1.45	mA
	$I_{IN(OFF)}$	4	All	$I_C = 500\text{ }\mu\text{A}, T_A = 70^\circ\text{C}$	50	65	—	$\mu\text{A}$
Input Voltage	$V_{IN(ON)}$	5	ULN2023A/L	$V_{CE} = 2.0\text{ V}, I_C = 200\text{ mA}$	—	—	2.4	V
				$V_{CE} = 2.0\text{ V}, I_C = 250\text{ mA}$	—	—	2.7	V
				$V_{CE} = 2.0\text{ V}, I_C = 300\text{ mA}$	—	—	3.0	V
		5	ULN2024A/L	$V_{CE} = 2.0\text{ V}, I_C = 125\text{ mA}$	—	—	5.0	V
				$V_{CE} = 2.0\text{ V}, I_C = 200\text{ mA}$	—	—	6.0	V
				$V_{CE} = 2.0\text{ V}, I_C = 275\text{ mA}$	—	—	7.0	V
				$V_{CE} = 2.0\text{ V}, I_C = 350\text{ mA}$	—	—	8.0	V
Input Capacitance	$C_{IN}$	—	All		—	15	25	pF
Turn-On Delay	$t_{PLH}$	8	All	$0.5 E_{IN}$ to $0.5 E_{OUT}$	—	0.25	1.0	$\mu\text{s}$
Turn-Off Delay	$t_{PHL}$	8	All	$0.5 E_{IN}$ to $0.5 E_{OUT}$	—	0.25	1.0	$\mu\text{s}$
Clamp Diode Leakage Current	$I_R$	6	All	$V_R = 95\text{ V}, T_A = 25^\circ\text{C}$	—	—	50	$\mu\text{A}$
				$V_R = 95\text{ V}, T_A = 70^\circ\text{C}$	—	—	100	$\mu\text{A}$
Clamp Diode Forward Voltage	$V_F$	7	All	$I_F = 350\text{ mA}$	—	1.7	2.0	V

Complete part number includes suffix to identify package style: A = DIP, L = SOIC.

# 2003 THRU 2024 HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON ARRAYS

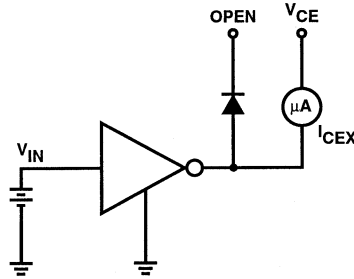
## TEST FIGURES

FIGURE 1A



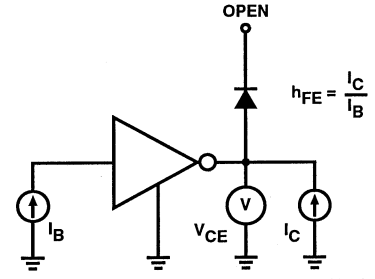
Dwg. No. A-9729A

FIGURE 1B



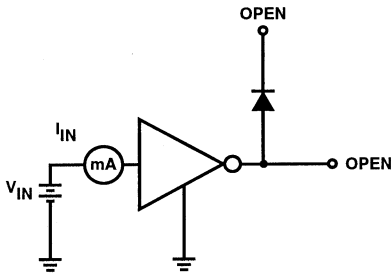
Dwg. No. A-9730A

FIGURE 2



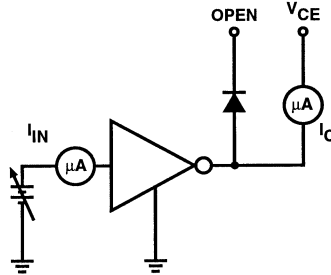
Dwg. No. A-9731A

FIGURE 3



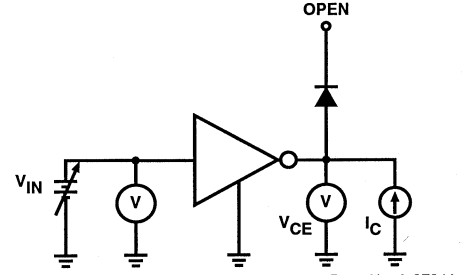
Dwg. No. A-9732A

FIGURE 4



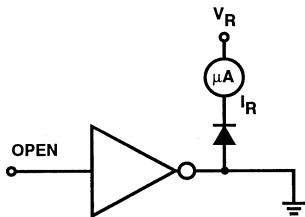
Dwg. No. A-9733A

FIGURE 5



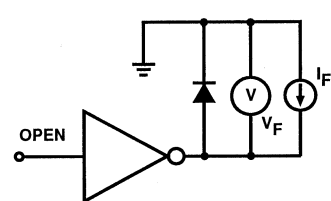
Dwg. No. A-9734A

FIGURE 6



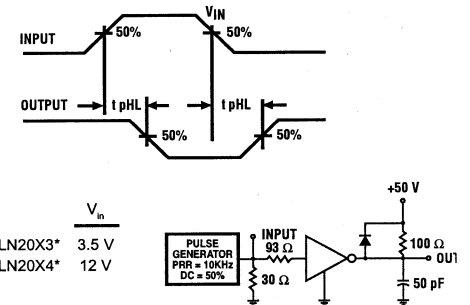
Dwg. No. A-9735A

FIGURE 7



Dwg. No. A-9736A

FIGURE 8

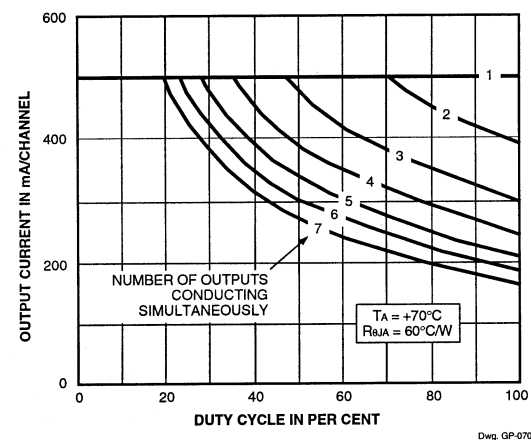


\* Complete part number includes a final letter to indicate package.

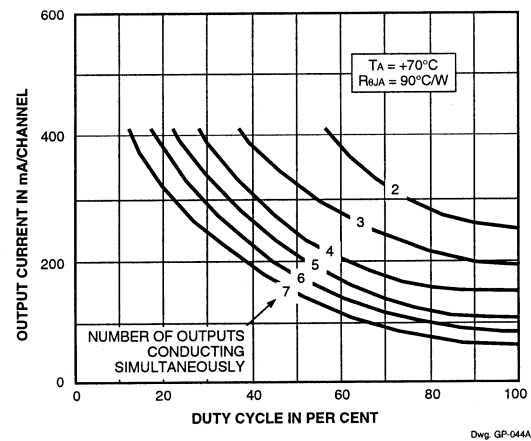
X = Digit to identify specific device. Specification shown applies to family of devices with remaining digits as shown.

# 2003 THRU 2024 HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON ARRAYS

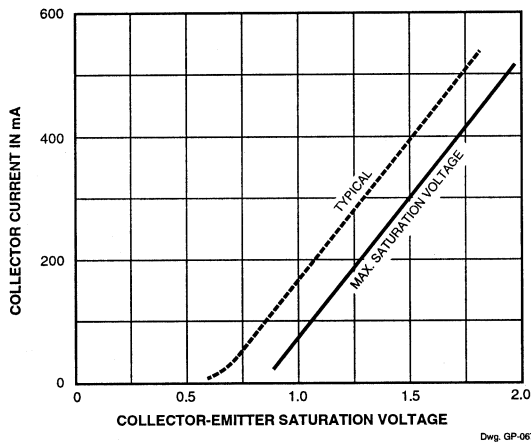
ALLOWABLE COLLECTOR CURRENT  
AS A FUNCTION OF DUTY CYCLE  
(Dual In-line-Packaged Devices, Suffix 'A')



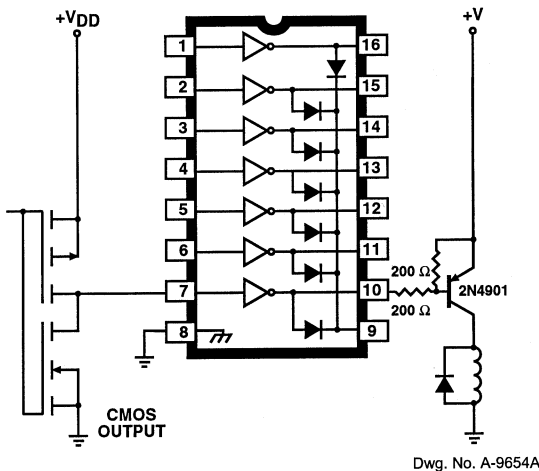
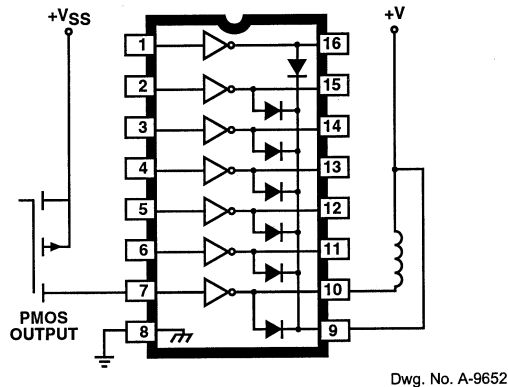
(Small-Outline-Packaged Devices, Suffix 'L')



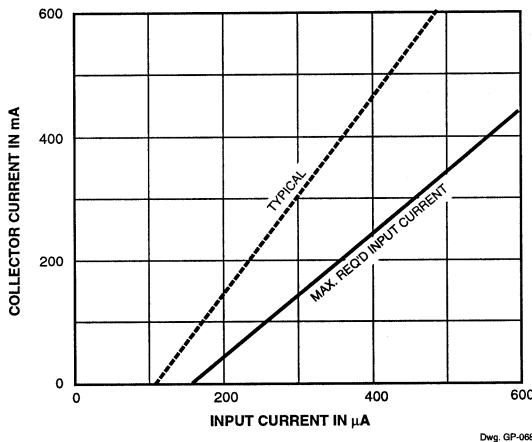
SATURATION VOLTAGE  
AS A FUNCTION OF COLLECTOR CURRENT



TYPICAL APPLICATIONS

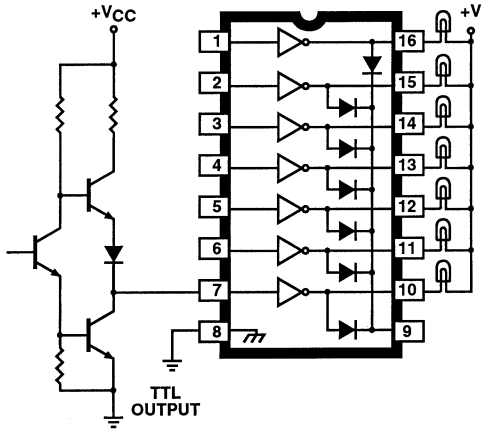


COLLECTOR CURRENT AS A  
FUNCTION OF INPUT CURRENT

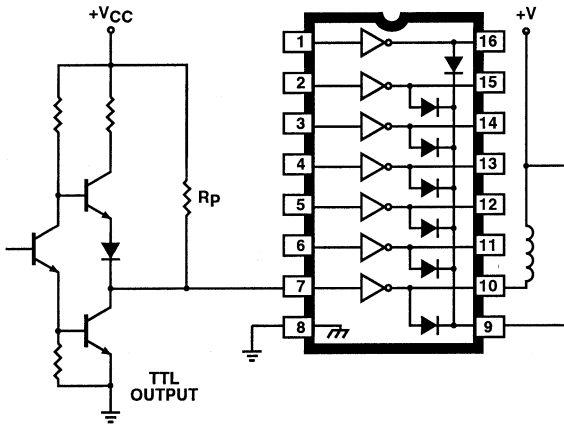


# 2003 THRU 2024 HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON ARRAYS

## TYPICAL APPLICATIONS



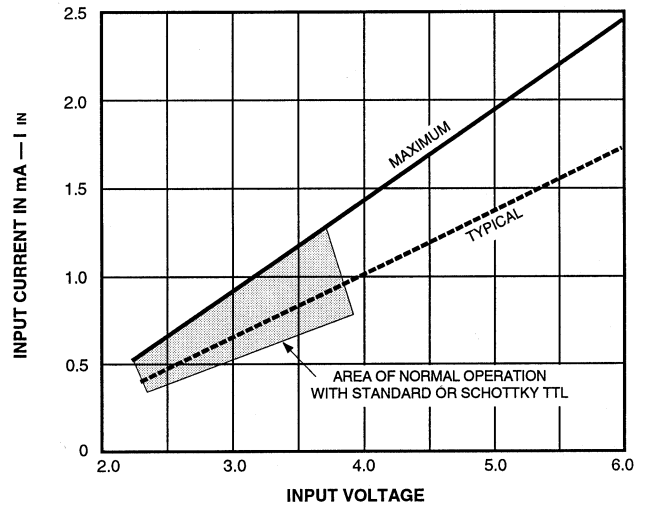
Dwg. No. A-9653A



Dwg. No. A-10,175

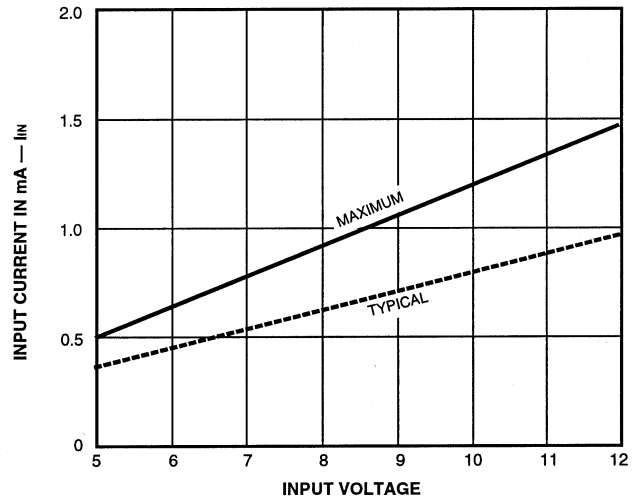
## INPUT CURRENT AS A FUNCTION OF INPUT VOLTAGE

Types ULN2003A, ULN2003L, ULN2023A, and  
ULN2023L



Dwg. GP-069

Types ULN2004A, ULN2004L, ULN2024A, and  
ULN2024L



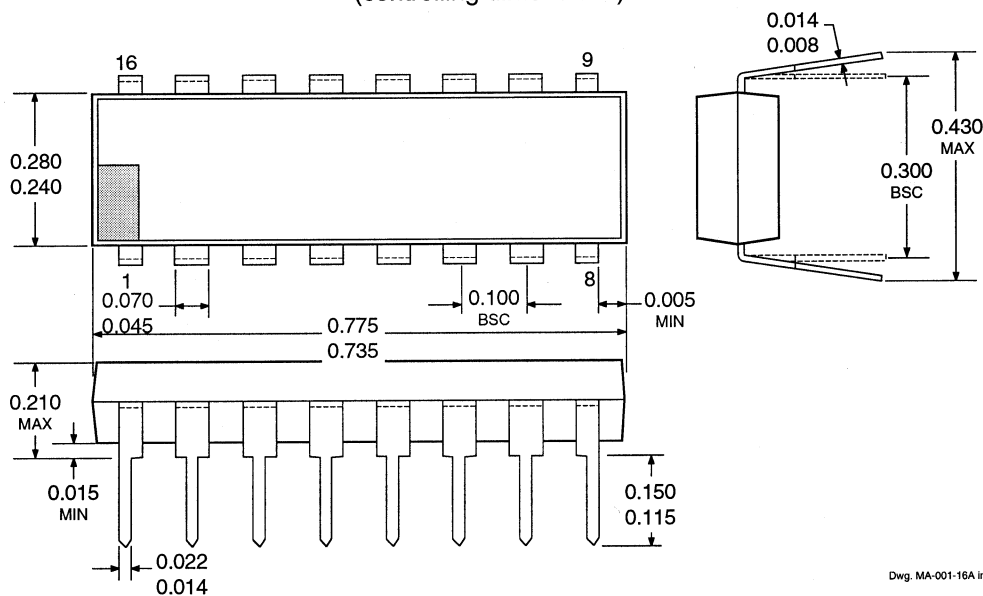
Dwg. GP-069-1



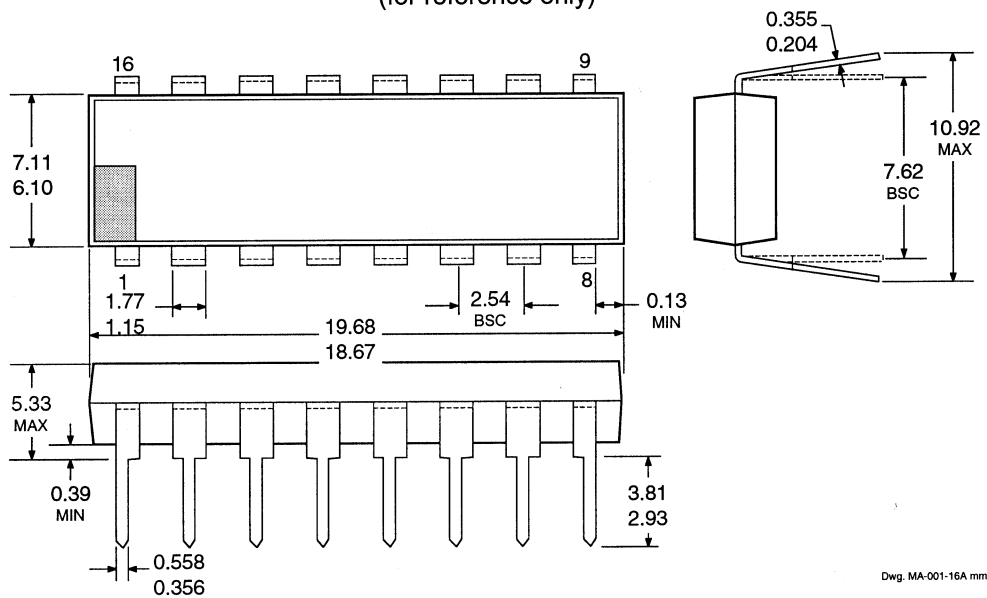
**2003 THRU 2024**  
**HIGH-VOLTAGE,**  
**HIGH-CURRENT**  
**DARLINGTON ARRAYS**

**PACKAGE DESIGNATOR "A"**

Dimensions in Inches  
 (controlling dimensions)



Dimension in Millimeters  
 (for reference only)

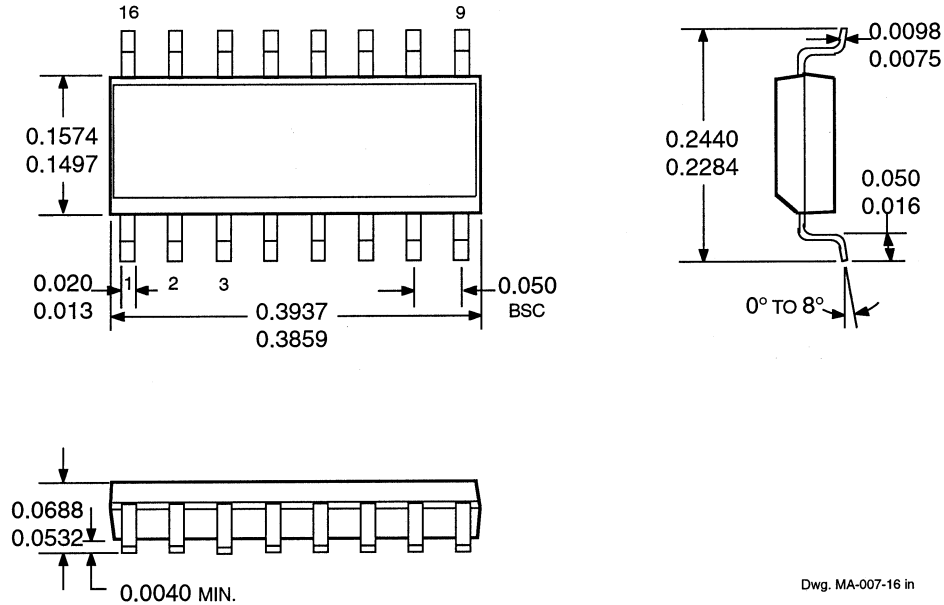


- NOTES: 1. Leads 1, 8, 9, and 16 may be half leads at vendor's option.  
 2. Lead thickness is measured at seating plane or below.  
 3. Lead spacing tolerance is non-cumulative.  
 4. Exact body and lead configuration at vendor's option within limits shown.

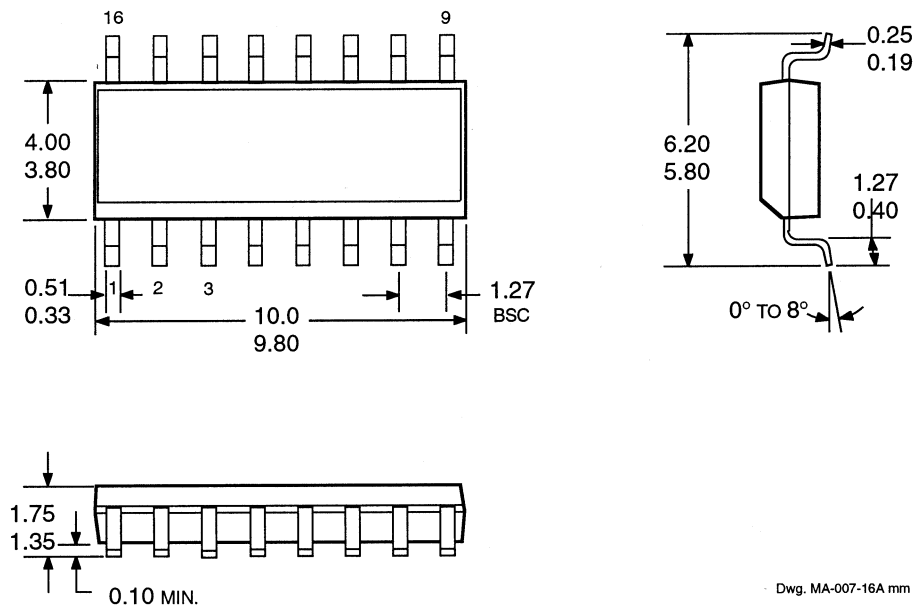
**2003 THRU 2024**  
**HIGH-VOLTAGE,**  
**HIGH-CURRENT**  
**DARLINGTON ARRAYS**

**PACKAGE DESIGNATOR "L"**

Dimensions in Inches  
(for reference only)



Dimension in Millimeters  
(controlling dimensions)



- NOTES: 1. Lead spacing tolerance is non-cumulative.  
2. Exact body and lead configuration at vendor's option within limits shown.

Student's name & ID: Tan Cheung

Student's name & ID: Michael Lee

Student's name & ID: Thomas Lo

Wai Nang Tsang

Project Title Housing Climate System

Date: 11/21/2001

Notes:

---



---



---



---

		POINTS
Proj. description & Purpose	(3 points max.)	3
Accomplishments/Progress to date	(3 points max.)	3
Analysis & Practical adjustments	(4 points max.)	4
Revised Schedule & assessment	(3 points max.)	3
Style, grammar, organization	(2 points max)	2
TOTAL		15

Grader's signature: zt

To: Prof. R.P. Kraft  
From: Tan Cheung  
Michael Lee  
Thomas Lo  
Wai Nang Tsang  
Subject: Project Interim Memo for Housing Climate System (a.k.a WIN2K1)  
Date: November 15, 2001

### **Project and Purpose**

Our project is to redesign the household climate control system. We are doing this to improve energy conservation and make a design with improved efficiency. We planned to implement fuzzy logic to improve upon current design. Our project will open and close windows when the <sup>temperature of the</sup> climate is too warm or too cold. We will check and maintain the temperature by using A/D conversion and interrupts. We use a ~~temperature~~ <sup>temperature</sup> sensor to sense the current temperature. We will simulate a cooling and heating system and a windows frame that opens and closes according to temperature. Our fuzzy logic will determine trends in temperature and operate in winter, summer, and spring/fall mode.

### **Progress Made**

We have bought and implemented a temperature-sensing unit onto the HC12 EVB. The temperature sensor gives a voltage between .2V to .4V. We needed to use an op-amp circuit to up the voltage to between 2V and 4V. We connect the sensor to the PD2 of the EVB, which is the A/D conversion port. The port takes the voltage and converts to a number, which we can use to determine temperature.

We have also worked on the LCD display. We maintain the same configuration as the magic eight ball lab. As the same in the magic eight ball lab, we have 4 choices for the user to choose from. User could control the windows opening, closing, displaying current temperature, and using smart mode. Basically, the LCD display and the keypad is just an interface between the user and the HC12. It allows the user to gain full control of the system. As for now, we have accomplished to have the LCD panel to display the menu and get all the user entered information, however, we are in the progress of working to interface with the other parts. In the OPEN mode and the CLOSE mode, our program will prompt the user to enter a value to determine how much the window is going to be open or closed. Once the user entered the SMART mode, the LCD panel will temporarily go into a stage that user don't have any access to it, until the # key is pressed, which means the end of the SMART mode.

Due to the ability to control its speed and rotations, a stepper motor was chosen to control the movement of the window. This is a 12V motor, with a step of 3.6 degrees. To control the motor, an IC package containing Darlington arrays were used. To rotate the motor, the 4 separate coils need to be engaged in a sequence. But because of the limited IO ports on the EVB, it was better to use a two-coil excitation scheme, and use two IO data bits to control the motor instead of four. The disadvantage of this is that we cannot half step the motor. In addition, the motor does not rotate as smoothly and consumes more power. Although, it does produce more torque. Since we will be using this to control the movement of a large, heavy window, this arrangement will suffice.


2

✓

We have built the window frame for our project to simulate the opening and closing due to changing climates. The frame consists of plywood and a sliding plexiglass pane. The stepper motor slides the window up and down.

### **Practical Implications**

We did not foresee the difficulty in implementing all the parts of our project onto one program. We are having trouble interfacing the different subsystems on the whole. We plan to spend more time on interfacing and making everything work smoothly. The fuzzy logic part of our project has been pushed to a later date because we plan to do additional research on how to best implement this. If we have additional time, we will implement light sensing abilities into our project. We will use photo resistors to determine the light level and turn on or off light automatically. We also might require a clock or timer into our project to make it smarter. We will use the interrupt as a basis for doing this. We plan to use a hair dryer and a small fan to simulate heating and cooling, therefore we will need these components.



Task	October	November	December
Wink1	10/17	11/15	12/9
LCD Panel	10/17	11/1	
Keypad	10/17	11/1	
Motor Control			
Thermostat	10/22	11/1	
Fuzzy Logic			
Testing & Debugging			
Window Frame			
Interfacing	10/28	10/31	

**To:** Prof. R. P. Kraft  
**From:** Tan Cheung  
Michael Lee  
Thomas Lo  
Wai Nang Tsang  
**Subject:** Project Proposal for Household Climate Control System (a.k.a WIN2K1)  
**Date:** October 11, 2001

### **Summary of Project**

With the ever-increasing climate control systems in households, we feel that these systems are inefficient with energy conservation. We wish to improve upon current design, to improve functionality resulting in better energy efficiency and ease of use. We see opportunity in improving upon these systems by adding more enhancements, i.e. fuzzy logic, LCD user interface. Basically, we are planning to redesign the home climate control so that by using the Timer Output Compare, we could control motors for opening and closing windows, check/maintain temperature and/or humidity using A/D converters. Our design would include fuzzy logic to automate the climate controller looking at the user's usage patterns so that the climate would be more suitable to him/her.

---

### **Problem Statement**

There is a need for our product because of the significant impact of California energy crisis and the dwindling supply of fossil fuel. This problem affects the general public and will deplete the supply of energy for future generations. As a result, by using our products, homeowners will come to appreciate the savings in energy costs.

### **Proposed Project, Goals, and Approach**

Our project, if programmed properly, can be more energy efficient by reducing the usage of air conditioning and heat by analyzing usage patterns, and calculating energy usage patterns. Alternatives that can be considered would be alternative fuels, for instance, nuclear power, solar energy, or hydropower, which may come in the future. Users will benefit from this design by being able to live comfortably, yet conserve the few precious reserves of energy left.

### **Plan of Activities with Deadline**

Our project would require the M68HC12 EVB, an LCD display, a Fuzzy Logic Controller, Temperature Sensitive Transistors, a Real Time Clock/Calendar Chip, relays, and motors.

### **Evaluation**

The project can be seen to solve the problems mentioned above simply by comparing energy usage, and by measuring the level of comfort of the user.

Week of 10/14	Motor	Fuzzy Logic	Thermostat	
10/21	Motor	Fuzzy Logic	Thermostat	
10/28	Motor	Fuzzy Logic	Thermostat	
11/4	Keypad	Fuzzy Logic	LCD	
11/11	Keypad	Report	LCD	
11/18	Keypad		LCD	
11/25	Testing & Debugging			
12/2	Testing, Debugging			

Motor - Wai Nang Tsang, Thomas Lo

Fuzzy logic - Thomas Lo, Wai Nang Tsang, Michael Lee, Tan cheung

Thermostat. - Tan cheung, Michael Lee

LCD. - Wai Nang Tsang, Thomas Lo

Keypad. - Tan cheung, Michael Lee.



Task	October	November	December
All Projects	10/12	11/5 Intern Demo	12/9 Final Demo
LED Panel		11/4	11/18
Keypad		11/4	11/18
Motor Control	10/14	10/28	
Thermostat	10/14	10/28	
Fuzzy Logic	10/14	11/4	
Testing & Debugging		11/15	12/9