Humidity Sensor & DC Fan Control

06 September 2006 Application Note (DKAN0007A)

Features

- DC fan control
- Measure capacitive sensor
- 555 Timer usage
- Microchip 12F683

Introduction

This application note covers a simple humidity sensing DC fan controller. Many sensors use capacitance as their form of measurement. A simple means of measuring capacitance is needed for these sensors. A short discussion on how to implement one type of measurement follows. This measurement is then utilized in the operation of a DC fan.

Background

Previous experience with a Microchip microcontroller using C programming is assumed. This circuit uses the PIC12F683 with an internal 4MHz clock. A basic knowledge of 555 timers would be helpful. A Texas Instruments TLC555 is used. Together these two chips, along with the Humirel HS1101 humidity sensor, will make up the major components of the circuit. The HS1101 is a capacitive type sensor that will operate from 1-99% relative humidity.

Application

The first subject is the 555 timer and how it will work in the circuit. The timer will use the capacitance of the HS1101 sensor to create a square waveform. To do this, there are several discrete components that need to be calculated. Three resistors are used to gain the desired effect, but only two were calculated. The square wave has an on cycle and an off cycle which gives the duty cycle of the signal. To achieve a duty cycle close to 51%, calculations for R1 and R2 will need to be made. Since C is already known, the HS1101 sensor acts as a capacitor, calculations are straight-forward. A base of 6500Hz is used when the sensor is 180pF or 55% relative humidity, thus calculations are:

F = 1/T = 6500Hz at 55% RH(C=180pF) F=Frequency T=Time period
-Offtime = 49%(T) = 0.00007538 = 0.693*R2*180pF, giving R2 = ~604K
604K is used, since it is a common value.
-Ontime = 51%(T) = 0.693*(R1+604K)*180pF, giving R1 = ~25K
25.5K is used, since it is a common value.

Figure 1 shows how to setup the 555 Timer and where to place the components.

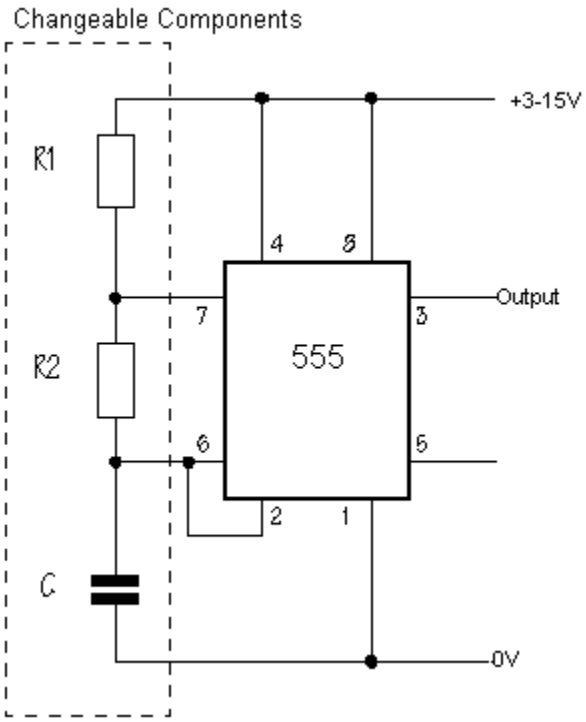

Figure 1: 555 Timer components

Power for the timer and all other components is 5Volts. A capacitor can be placed on pin 5 of the timer for stability, but is not required. The last component on the timer is a 1K resistor placed on the output for current regulation and spurious signals.
Setting up the 555 timer is important, because the microcontroller uses this signal for implementing the sleep function to conserve power.
The microcontroller utilizes its internal clock and peripheral interrupts for the calculations needed. The registers to set are PIE, T1CON, OPTION_REG, ANSEL, and PEIE. These registers set up the microcontroller to count the incoming signal transitions and calculate the frequency being generated. The number of low to high transitions in one second is the how the frequency is calculated (frequency = 1/number of transitions). When the microcontroller is placed in sleep mode, the same input signal is used to increment the internal timer using the low to high transition, thus the need for the timer signal on two pins. As the timer overflows, a flag is set, and the microcontroller wakes up from sleep to do another calculation.

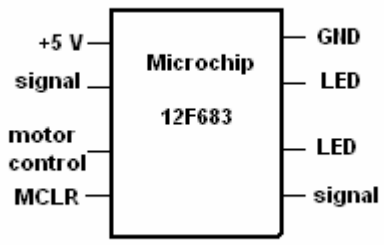

Figure 2: Microchip 12F683 I/O lines

A MOSFET takes the load away from the micro and keeps the power drain near the source. In the schematic, there is a 47uF cap at the connector for the motor. This is because several alterations were tested that could not handle the inrush current draw from the motor start-up, and this caused a failure in the microcontroller. It is important to have a large cap at the connector.

Capabilities, Limitations, and Alterations

Since the 555 timer can be very flexible in implementation, any number of combinations can be practically used. The duty cycle does not need to be 50%, because the microcontroller does not care about the duty cycle. It only cares about the change from low to high on the pulsed waveform. Frequency is our main concern with the timer. Other more power efficient circuits can be implemented with the timer.
There are more direct, but complex, ways to measure capacitance. These other methods rely on a more in depth knowledge of components and will vary depending on the setup.
This circuit can be produced for larger or smaller fans based on the FET used.
An advanced circuit can be created using one of the pins of the microcontroller and a potentiometer to set the desired humidity level. The possibility is also available to run a small LCD for output and level settings, but this would need a larger microcontroller.

Conclusion

This simple circuit uses a 555 Timer for measuring the capacity of a humidity sensor. The microcontroller reads the number of incoming pulses and translates this into frequency. Depending on the application, a pre-stored value is compared to the frequency, and the motor is turned on or off. Humidity level is set based on the pre-stored value. The application of this circuit is for a humidor or other environmentally controlled small area.

Additional Information

There are many websites that have information on 555 timers; some are more helpful than others. Here are a few that were used for researching this note.
http://www.uoguelph.ca/~antoon/gadgets/555/555.html
http://www.doctronics.co.uk/555.htm
http://home.cogeco.ca/~rpaisley4/LM555.html

Appendix A: Complete Source Code

```
//Program used to check the humidity of surrounding space if the humidity
//is too low then a motor will kick on and humidify the space.  This could
// be used with the opposite intentions (to dry an area if humidity is too high)
//
#include <12F683.h>                    //this file comes with PICC C Compiler
#include <my12F683.h>                  //This is a file I created
```

```
//****************Subroutine declarations**********************************
void Wait_4ms(unsigned int);

//****************Global Variables*****************************************
int MIN_Humidity;                   //this will set the minimum humidity level
int TMR_CNT;                        //holds the value of timer 0

//********************Subroutines*****************************************
//at X=250 you have one second time delay      if main clock = 4Mhz

void Wait_4ms(unsigned int X)       //this is merely a wait loop
{
    unsigned int Y;
    unsigned int P;
    do
    {
        Y = 4;
        do
        {
            P = 166;
            do
            {
                P -= 1;
            } while (P > 0);
            Y -= 1;
        } while (Y > 0);
        X -= 1;
    } while (X > 0);
}

//******************Main prog*************************************

void main (void)
{

//**********************setup*************************************************

    PIE1=0x01;                      //setting interrupt enables
    T1CON=0x06;                     //setup timer for internal clock/sleep operations
    OPTION_REG=0x65;                //sleep function will use 555timer to wake up
    ANSEL=0x00;                     //no analog channels
    PEIE=1;                         //peripheral interrupt enable
    MIN_Humidity = 100;             //for min humidity ~64%
                                    // this will change a little with each circuit
    TMR_CNT=0x0000;                 //resetting just to make sure
    GPIO = 0x03;                    // set I/O port to GP0,GP1—on, the rest--off
```

```
    TRISA=0xAC;                     // setting pins to output or input
    WPU=0xAC;                       //turning on WP5 and WP2 weak pull-ups

//**********start main loop**********

    do
    {
        Wait_4ms(2);                //a short wait for stablization
        RA0=0;                      // turn on led
        TMR0=0x00;                  //clear timer for counting
        Wait_4ms(250);              //led shows the data collection time ~1sec
        TMR_CNT=0x0000;             // resetting ---another precaution
        TMR_CNT=TMR0;               //store the number of edges
        RA0=1;                      //turn off LED
        if (TMR_CNT >= MIN_Humidity)  //testing the count for low humidity
        {
              RA4=1;                //turn motor on
        }
        if (TMR_CNT <= MIN_Humidity)
        {
              RA4=0;                //motor off
        }

//**********Timer1 setup for sleep**********
        TMR1H=0x00;                 //these are precautionary
        TMR1L=0x00;
        TMR1IF=0;
        RA1 = 0;                    //LED signal for sleep
                                    //this is not needed but is nice for debug
        Wait_4ms(4);                //          |
        RA1 = 1;                    //          V
        TMR1ON=1;                   //timer 1 on for wake-up
        SLEEP();

//***wake up from sleep after Timer 0 interupts**********

        TMR1ON=0;                   //timer 1 off while in main loop
    } while(1>0);                   //end do while loop (main loop)
}                                   //end of main program
```

Appendix B: Secondary Header File (my12F683.h)

```
//#device PIC12F683
//#nolist

//This file uses the actual names that are in the datasheet
//which helps in programming and manipulation of the micro
```

```
//there are more names than is required

//************************************GPIO*********************

#BYTE GPIO = 0x05
#BIT RA0  =  0x05.0
#BIT RA1  =  0x05.1
#BIT RA2  =  0x05.2                  // these are not actual datasheet
#BIT RA3  =  0x05.3                  //names but are short
#BIT RA4  =  0x05.4                  // and easy to use
#BIT RA5  =  0x05.5

//*********************************TRISA*********************
#byte TRISA   = 0x85

//*********************************WPU**********************
#byte WPU = 0x95

//*********************************INTCON*********************
#byte INTCON = 0x0B
#BIT GPIF  = 0x0B.0
#BIT INTF  = 0x0B.1
#BIT T0IF  =  0x0B.2
#BIT GPIE  = 0x0B.3
#BIT INTE  = 0x0B.4
#BIT TOIE  = 0x0B.5
#BIT PEIE  = 0x0B.6
#BIT GIE   =  0x0B.7

//*********************************T1CON*********************
#BYTE  T1CON = 0x10
#BIT TMR1ON  =   0x10.0
#BIT TMR1CS  =   0x10.1
#BIT T1SYNC  =  0x10.2
#BIT T1OSCEN  =  0x10.3
#BIT T1CKPSO  =  0x10.4
#BIT T1CKPS1  =  0x10.5
#BIT TMR1GE  =   0x10.6
#BIT T1GINV   =   0x10.7

//*****************************CMCON1*********************
#BIT CMSYNC  =  0x1A.0
#BIT T1GSS  =   0x1A.1

//*****************************PIR1*********************
#BIT TMR1IF  = 0x0C.0
#BIT TMR2IF  = 0x0C.1
```

```
//**************************PIE1*****************************
#define PIE1    *0x8C

//**************************OSCCON***************************
#define OSCCON    *0x8F

//**************************ANSEL****************************
#define ANSEL    *0x9F

#BYTE TMR1L = 0x0E
#BYTE TMR1H = 0x0F
#BYTE PR2 = 0x92
#BYTE  T2CON = 0x12
#byte TMR2 = 0x11
#byte TMR0 = 0x01
#BIT TMR2ON = 0x12.2

//*****************OPTION_REG**********************************
#BYTE OPTION_REG = 0x81
#BIT PS0  =    0x81.0
#BIT PS1  =    0x81.1
#BIT PS2  =   0x81.2
#BIT PSA  =   0x81.3
#BIT T0SE  = 0x81.4
#BIT T0CS  = 0x81.5
#BIT INTEDG  =0x81.6
#BIT GPPU   =    0x81.7
```

Appendix C: General Schematic

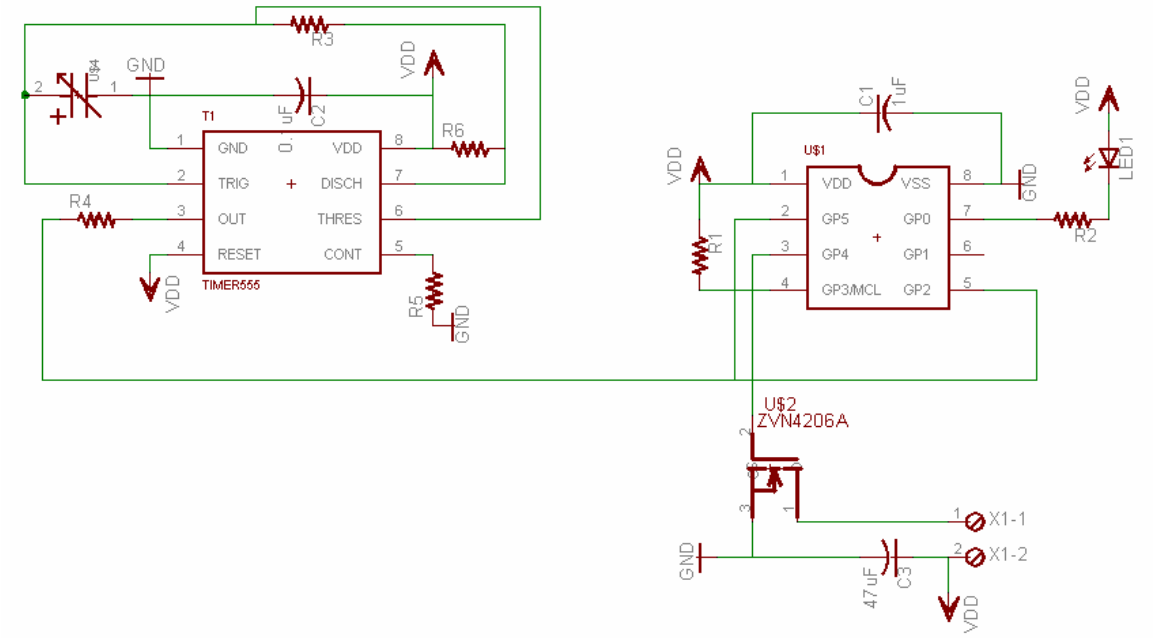

Figure 3: Schematic

Appendix D: Parts List

Part	Part Number
Microcontroller	PIC12F683-I/P-ND
555 Timer	296-1857-5-ND
Humidity Sensor	HS1101-ND
MOSFET U$2	ZVN4206A-ND
R1	100KXBK-ND
R2	ERO-S2PHF3320-ND
LED1	404-1106-ND
C1	493-1333-ND
C3	P12923-ND
R3	604KXBK-ND
R4	1.00KXBK-ND
R5	909KXBK-ND
R6	25.5KXBK-ND
C2	493-1095-ND
Terminal (2)	277-1273-ND

Disclaimer

Digi-Key offers its Technical Assistance and Design Support Services as a convenience to Digi-Key customers. Digi-Key Technical Assistance and Design Support Services personnel strive to provide useful information regarding Digi-Key products. DIGI-KEY DOES NOT GUARANTEE THAT ANY INFORMATION OR RECOMMENDATION PROVIDED IS ACCURATE, COMPLETE, OR CORRECT, AND DIGI-KEY SHALL HAVE NO RESPONSIBILITY OR LIABILITY WHATSOEVER IN CONNECTION WITH ANY INFORMATION OR RECOMMENDATION PROVIDED, OR THE CUSTOMER'S RELIANCE ON SUCH INFORMATION OR RECOMMENDATION. THE CUSTOMER IS SOLELY RESPONSIBLE FOR ANALYZING AND DETERMINING THE APPROPRIATENESS OF ANY INFORMATION OR RECOMMENDATION PROVIDED BY DIGI-KEY TECHNICAL ASSISTANCE AND DESIGN SUPPORT SERVICES PERSONNEL, AND ANY RELIANCE ON SUCH INFORMATION OR RECOMMENDATION IS AT THE CUSTOMER'S SOLE RISK AND DISCRETION. ACCORDINGLY, THE CUSTOMER SHALL RELEASE AND HOLD DIGI-KEY HARMLESS FROM AND AGAINST ANY AND ALL LOSS, LIABILITY, AND DAMAGE INCURRED BY THE CUSTOMER OR ANY THIRD PARTY AS A RESULT OF ANY INFORMATION OR RECOMMENDATION PROVIDED TO THE CUSTOMER OR THE CUSTOMER'S RELIANCE ON SUCH INFORMATION OR RECOMMENDATION.