Interfacing with a Honeywell HIH-6130/6131 Humidity Temperature Sensor (Arduino)

copyright, Peter H. Anderson, Baltimore, MD, Oct 20, '11

This application note presents the use of a Honeywell HIH-6130 using an Arduino. The code is pretty straight forward and hopefully this will enable users to implement using other platforms.

The HIH-6130 and HIH-6131 are available at Mouser. I mounted mine on an SOIC to DIP adaptor suitable for use with a solderless breadboard in developing this note.

References.

I found these on Mouser's website. I hate to burden a commercial site and thus I have temporarily copied them to my site. If anyone find's them on a Honeywell site, please let me know.

- I2C Instructions
- Entering and Using the Command Mode
- Using Alarms

Program HIH_6130_1 - Measurement of RH and T

```
// HIH_6130_1  - Arduino
//
// Arduino                 HIH-6130
// SCL (Analog 5) ------- SCL (term 3)
// SDA (Analog 4) ------- SDA (term 4)
//
// Note 2.2K pullups to 5 VDC on both SDA and SCL
//
// Pin4 ----------------- Vdd (term 8)
//
// Illustrates how to measure relative humidity and temperature.
//
// copyright, Peter H Anderson, Baltimore, MD, Nov, '11
// You may use it, but please give credit.

#include <Wire.h> //I2C library

byte fetch_humidity_temperature(unsigned int *p_Humidity, unsigned int *p_Temperature);
void print_float(float f, int num_digits);

#define TRUE 1
#define FALSE 0

void setup(void)
{
   Serial.begin(9600);
   Wire.begin();
   pinMode(4, OUTPUT);
   digitalWrite(4, HIGH); // this turns on the HIH3610
   delay(5000);
   Serial.println(">>>>>>>>>>>>>>>>>>>>>>>>");  // just to be sure things are working
}

void loop(void)
{
   byte _status;
   unsigned int H_dat, T_dat;
   float RH, T_C;

   while(1)
   {
      _status = fetch_humidity_temperature(&H_dat, &T_dat);
      switch(_status)
      {
          case 0:  Serial.println("Normal.");
                   break;
          case 1:  Serial.println("Stale Data.");
                   break;
          case 2:  Serial.println("In command mode.");
                   break;
          default: Serial.println("Diagnostic.");
                   break;
      }

      RH = (float) H_dat * 6.10e-3;
      T_C = (float) T_dat * 1.007e-2 - 40.0;

      print_float(RH, 1);
      Serial.print("  ");
      print_float(T_C, 2);
      Serial.println();
      delay(1000);
   }
}

byte fetch_humidity_temperature(unsigned int *p_H_dat, unsigned int *p_T_dat)
{
      byte address, Hum_H, Hum_L, Temp_H, Temp_L, _status;
      unsigned int H_dat, T_dat;
      address = 0x27;;
      Wire.beginTransmission(address);
      Wire.endTransmission();
      delay(100);

      Wire.requestFrom((int)address, (int) 4);
      Hum_H = Wire.receive();
      Hum_L = Wire.receive();
      Temp_H = Wire.receive();
      Temp_L = Wire.receive();
      Wire.endTransmission();

      _status = (Hum_H >> 6) & 0x03;
      Hum_H = Hum_H & 0x3f;
      H_dat = (((unsigned int)Hum_H) << 8) | Hum_L;
      T_dat = (((unsigned int)Temp_H) << 8) | Temp_L;
      T_dat = T_dat / 4;
      *p_H_dat = H_dat;
      *p_T_dat = T_dat;
      return(_status);
}

void print_float(float f, int num_digits)
{
    int f_int;
    int pows_of_ten[4] = {1, 10, 100, 1000};
    int multiplier, whole, fract, d, n;

    multiplier = pows_of_ten[num_digits];
    if (f < 0.0)
    {
        f = -f;
        Serial.print("-");
    }
    whole = (int) f;
    fract = (int) (multiplier * (f - (float)whole));

    Serial.print(whole);
    Serial.print(".");

    for (n=num_digits-1; n>=0; n--) // print each digit with no leading zero suppression
    {
         d = fract / pows_of_ten[n];
         Serial.print(d);
        fract = fract % pows_of_ten[n];
    }
}
```

Program HIH_6130_2 - Command Mode

This routine illustrates how to enter the command mode, read from the HIH-6130 EEPROM, write to EEPROM, configure the alarm levels, read and display the alarm levels and write and read customer ID information. The device is then returned to normal operation.

This routine does not illustrate use of the Cust_Config Register at EEPROM location 0x1C. This allows for modifying the startup speed (3 or 10 ms), the alarm polarity and output configuration (push-pull or open drain) for both the high and low alarms and the I2C address. See Using Alarms, Table 3.

```
// HIH_6130_2  - Arduino
//
// Arduino                HIH-6130
//
// Digital 4 ------------ Vdd (term 8) // Note that the Arduino provides power to the device.
//
// SCL (Analog 5) ------- SCL (term 3)
// SDA (Analog 4) ------- SDA (term 4)
//
// Note 2.2K pullups to _5 VDC on both SDA and SCL
//
// copyright, Peter H Anderson, Baltimore, MD, Oct 14, '11
// You may use it, but please give credit.
//

#include <Wire.h> //I2C library

void write_word(byte command, unsigned int dat);
unsigned int read_word(byte command);
void write_alarm(byte command, float huma_alm);
float read_alarm(byte command);


#define TRUE 1
#define FALSE 0

void setup(void)
{
    Serial.begin(9600);
    Wire.begin();
    pinMode(4, OUTPUT);
    digitalWrite(4, LOW);
    delay(5000);
    Serial.println(">>>>>>>>>>>>>>>>>>>>>>>>");  // just to be sure things are working
}

void loop(void)
{
    byte n;
    unsigned int w;
    float v;

    digitalWrite(4, HIGH);  // turn on power
    write_word(0xa0, 0x0000);  // and enter command mode within 10 ms

    for (n=0; n<0x20; n++)
    {
        w = read_word(n);
        Serial.print(n, HEX);
        Serial.print("  ");
        print_hex(w, 16);
        Serial.println();
    }
```

```
     write_alarm(0x58, 80.0);  // high alarm on
     write_alarm(0x59, 75.0);  // high alarm off

     write_alarm(0x5a, 33.0);  // low alarm on
     write_alarm(0x5b, 40.0);  // low alarm off

     write_word(0x5e, 0xa5a5); // write data to customer ID locations
     write_word(0X5f, 0x5a5a);

     Serial.println(".......");

     for (n=0x18; n<0x20; n++)
     {
         w = read_word(n);
         Serial.print(n, HEX);
         Serial.print("  ");
         print_hex(w, 16);
         Serial.println();
     }

     Serial.println("............................");

     for (n=0x18; n<=0x1b; n++)
     {
         v = read_alarm(n);
         Serial.print(n, HEX);
         Serial.print("  ");
         print_float(v, 2);
         Serial.println();
     }

     write_word(0x80, 0x0000);  // go to normal operation

     Serial.println("Done");

     while(1)
     {
     }
}

void write_word(byte command, unsigned int dat)
{
      byte H, L;
      H = dat >> 8;
      L = dat & 0xff;

      Wire.beginTransmission(0x27);
      Wire.send(command);
      Wire.send(H);
      Wire.send(L);
      Wire.endTransmission();
      delay(15);
}

unsigned int read_word(byte command)
{
     byte high, low, response_byte;
     unsigned int w;

     write_word(command, 0x0000);
     Wire.requestFrom((int) 0x27, (int) 3);
     response_byte = Wire.receive();
     high = Wire.receive();
     low = Wire.receive();
     Wire.endTransmission();
     w = high;
     w = w *256 + low;
     return(w);
}

void write_alarm(byte command, float huma_alm)
```

```
{
    unsigned int w;
    w = (unsigned int)(huma_alm * 163.83);
    write_word(command, w);
}

float read_alarm(byte command)
{
    unsigned int w;
    float v;

    w = read_word(command);
    v = ((float) w) * 6.103e-3;
    return(v);
}

void print_hex(int v, int num_places)
{
    int mask=0, n, num_nibbles, digit;

    for (n=1; n<=num_places; n++)
    {
        mask = (mask << 1) | 0x0001;
    }
    v = v & mask; // truncate v to specified number of places

    num_nibbles = num_places / 4;
    if ((num_places % 4) != 0)
    {
        ++num_nibbles;
    }

    do
    {
        digit = ((v >> (num_nibbles-1) * 4)) & 0x0f;
        Serial.print(digit, HEX);
    } while(--num_nibbles);

}

void print_float(float f, int num_digits)
{
    int f_int;
    int pows_of_ten[4] = {1, 10, 100, 1000};
    int multiplier, whole, fract, d, n;

    multiplier = pows_of_ten[num_digits];
    if (f < 0.0)
    {
        f = -f;
        Serial.print("-");
    }
    whole = (int) f;
    fract = (int) (multiplier * (f - (float)whole));

    Serial.print(whole);
    Serial.print(".");

    for (n=num_digits-1; n>=0; n--) // print each digit with no leading zero suppression
    {
         d = fract / pows_of_ten[n];
         Serial.print(d);
         fract = fract % pows_of_ten[n];
    }
}
```