```c
/**********************************************
 * File:    humidicon.h
 * Author:  Bryant Gonzaga
 * Date:    3/6/2018
 **********************************************/

#ifndef HUMIDICON_H_
#define HUMIDICON_H_

#include <iom128.h>

#define _BV(bit) (1 << (bit))

//*******************************************************************
**********
// Function : void SPI_humidicon_config (void)
// Date and version : version 1.0
// Target MCU : ATmega128A @ 16MHz
// Author :
// DESCRIPTION
// This function unselects the HumidIcon and configures it for
operation with
// an ATmega128A operated a 16 MHz. Pin PA0 of the ATmega128A is used
to select
// the HumidIcon. SPI for humidicon has a max
// slave clock frequency of 800 kHz
//
// Modified
//*******************************************************************
**********
void spi_humidicon_config();

//*******************************************************************
**********
// Function : unsigned char read_humidicon_byte(void)
// Date and version : version 1.0
// Target MCU : ATmega128A
// Author : Ken Short
// DESCRIPTION
// This function reads a data byte from the HumidIcon sensor and
returns it as
// an unsigned char. The function does not return until the SPI
transfer is
// completed. The function determines whether the SPI transfer is
complete
// by polling the appropriate SPI status flag.
//
// Modified
//*******************************************************************
**********
unsigned char read_humidicon_byte();

//*******************************************************************
**********
// Function : void read_humidicon (void)
// Date and version : version 1.0
```

```c
// Target MCU : ATmega128A
// Author :
// DESCRIPTION
// This function selects the Humidicon by asserting PA0. It then calls
// read_humidicon_byte() four times to read the temperature and
humidity
// information. Is assigns the values read to the global unsigned ints
humidicon_byte1,
// humidion_byte2, humidion_byte3, and humidion_byte4, respectively.
The
// function then deselects the HumidIcon.
//
// The function then extracts the fourteen bits corresponding to the
humidity
// information and stores them right justified in the global unsigned
int humidity_raw.
// Next if extracts the fourteen bits corresponding to the temperature
// information and stores them in the global unsigned int
temperature_raw. The function
// then returns
//
// Modified
//*********************************************************************
**********
void read_humidicon();

long int compute_scaled_rh(unsigned int rh);
long int compute_scaled_temp(unsigned int temp);

#endif /* HUMIDICON_H_ */
```