Cika Electrónica

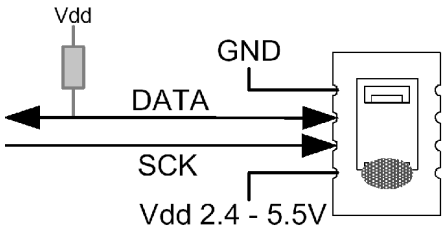
Nota de Aplicación: CAN-047
Título: **Sensores de Humedad y Temperatura SHT-71 con PIC**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	07/11/05	

Si bien existen numerosas alternativas para la medición de temperatura ambiente, la medición de la humedad relativa ambiente resulta ser algo complicada. En ambos casos, lograr una determinada precisión implica disponer de sensores caros y mucho cuidado en la sección analógica y la placa de circuito impreso; hecho que se ve potenciado si el sensor debe estar a una cierta distancia del circuito de procesamiento, y/o en ambientes hostiles. Los sensores combinados de humedad y temperatura ambiente SHT-71, desarrollados por la firma Sensirion, constan de un par de sensores y conversores A/D, circuitería de calibración, y compensación. La información es presentada por una interfaz serie, de modo que el sistema de medición se desentiende del tema ruidos, linealización analógica, calibración, e impedancias. Para más detalles, y código ejemplo en C, le recomendamos referirse a la CAN-028. En esta oportunidad, desarrollamos código para poder leer los sensores con PIC, bajando notablemente el costo para aplicaciones que requieran gran cantidad de sensores remotos.

Hardware

La conexión es sumamente simple, requiere dos pines de I/O, uno con capacidad tristate u open collector para los datos; el otro entrega el clock que controla el timing de esos datos, para ambos sentidos de la comunicación



Software

El fabricante nos provee código en C como para ser compilado en un procesador compatible MCS51, sin demasiado esfuerzo, es posible portarlo a assembler para cualquier otra arquitectura, tarea la cual hemos desarrollado para PIC, en la forma de una pequeña biblioteca de funciones.

El software a continuación se encarga de la lectura de la información provista por el sensor. No realizamos ningún tipo de procesamiento de la misma, tarea la cual dejamos a criterio del lector. En nuestro caso, y según veremos en la CAN-048, reportaremos este valor a un sistema central basado en Rabbit, el cual hará la linealización, corrección y presentación de los datos. Para tareas locales que no requieran gran precisión, el usuario puede procesar la información que colecta esta nota como un número entero de 16-bits, a su gusto.

Comenzamos por las funciones de lectura y escritura de un byte en el sensor.

```
; Envía un byte
; Dato en W
; Devuelve resultado en el flag de carry (C)
; C -> Error
s_write_byte:
       movwf dat                        ; guarda el dato
       movlw 8                          ; 8 bits
       movwf bitcntr
swbl1  rlf dat,f                        ; C = MSB (MSB sale primero)
       bcf SCK                          ; CLK = 0
```

```
        nop
        bcf DAT                         ; DATA = 0 (open-collector)
        bsf STATUS,RP0                  ; Bank 1
        bsf DATT                        ; pull-up
        btfss STATUS,C                  ; if C, DATA = pull-up
        bcf DATT                        ; sino DATA = 0
        bcf STATUS,RP0                  ; Bank 0
        bsf SCK                         ; CLK = 1
        decfsz bitcntr,f
        goto swbl1
        bcf SCK                         ; CLK = 0
        bsf STATUS,RP0                  ; Bank 1
        bsf DATT                        ; pull-up
        bcf STATUS,RP0                  ; Bank 0
        bsf SCK                         ; CLK = 1 (9º clk)
        rrf DATPORT,W                   ; check ack (el SHT pone DATA en 0)
        bcf SCK                         ; CLK = 0
        return

; Recibe un byte
; Devuelve el dato en W
s_read_byte:
        clrf dat                        ; DATA = 0
        movlw 8                         ; 8 bits
        movwf bitcntr
srbl1   bsf SCK                         ; CLK = 1
        nop
        rrf DATPORT,W                   ; lee bit en C
        rlf dat,f                       ; y lo lleva al registro (MSB primero)
        bcf SCK                         ; CLK = 0
        decfsz bitcntr,f
        goto srbl1
        bcf DAT                         ; DATA = 0 (open-collector)
        bsf STATUS,RP0                  ; Bank 1
        btfsc ACK                       ; si debe hacer ACK
        bcf DATT                        ; DATA=0 (ACK)
        bcf STATUS,RP0                  ; Bank 0
        bsf SCK                         ; CLK = 1 (9º clk)
        nop
        bcf SCK                         ; CLK = 0
        bsf STATUS,RP0                  ; Bank 1
        bsf DATT                        ; pull-up
        bcf STATUS,RP0                  ; Bank 0
        movf dat,W
        return
```

Como habrán deducido, la selección de los pines a emplear se realiza mediante macros, las cuales pueden redefinirse dentro del programa:

```
; DAT debe ser el LSB, de otro modo hay que modificar el programa
#define DAT    PORTC,0
#define DATPORT PORTC
#define DATT   TRISC,0
#define SCK    PORTC,1

;                           addr  command  r/w
#define STATUS_REG_W 0x06   ; 000   0011    0
#define STATUS_REG_R 0x07   ; 000   0011    1
#define MEASURE_TEMP 0x03   ; 000   0001    1
#define MEASURE_HUMI 0x05   ; 000   0010    1
#define SHT_RESET    0x1e   ; 000   1111    0

shtvars UDATA_SHR
dat     RES 1
bitcntr RES 1
value   RES 2
chksum  RES 1
flags   RES 1

#define ACK flags,0
```

Luego seguimos por el proceso de inicialización.

```
; reset: DATA=1 y 9 SCK, seguidos de transstart
;      _____________________________________________________         ________
; DATA:                                                     |_______|
;          _   _   _   _   _   _   _   _   _   _   _        ___     ___
; SCK : __| |__| |__| |__| |__| |__| |__| |__| |__| |______|   |___|   |______
; Se debe llamar al inicio o si se observan errores
s_connectionreset:
       bcf SCK                          ; CLK = 0
       movlw 9                          ; 9 SCK
       movwf bitcntr
scrl1  bsf SCK                          ; CLK = 1
       nop
       bcf SCK                          ; CLK = 0
       decfsz bitcntr,f
       goto scrl1
                                        ; sigue transmission start
;genera un transmission start
;       _____           ________
; DATA:      |_______|
;           ___     ___
; SCK : ___|   |___|   |______
s_transstart:
       bcf SCK                          ; CLK = 0
       nop
       bsf SCK                          ; CLK = 1
       nop
       bcf DAT                          ; DATA = 0 (open-collector)
       bsf STATUS,RP0                   ; Bank 1
       bcf DATT                         ; DATA=0
       bcf STATUS,RP0                   ; Bank 0
       bcf SCK                          ; CLK = 0
       nop
       bsf SCK                          ; CLK = 1
       bsf STATUS,RP0                   ; Bank 1
       bsf DATT                         ; pull-up
       bcf STATUS,RP0                   ; Bank 0
       bcf SCK                          ; CLK = 0
       return
```

A continuación, un juego de rutinas que nos permite iniciar una medición, interrogar al sensor el estado de la misma, y obtener el resultado.

```
; inicia medición de tmperatura
s_measure_t:
       call s_transstart                ; transmission start
       movlw MEASURE_TEMP
       goto s_write_byte                ; manda comando al sensor y retorna

; inicia medición de humedad
s_measure_rh:
       call s_transstart                ; transmission start
       movlw MEASURE_HUMI
       goto s_write_byte                ; manda comando al sensor y retorna

; Llamar hasta que regresa sin carry (C=0), o transcurre demasiado tiempo
s_measure_wait:
       rrf DATPORT,W                    ; lee dato en C
       return

; Obtiene resultados
s_measure_get:
       bsf ACK                          ; ACK
       call s_read_byte                 ; lee primer byte (MSB)
       movwf value+1
       call s_read_byte                 ; lee segundo byte (LSB)
       movwf value
       bcf ACK                          ; no ACK
       call s_read_byte                 ; lee checksum (8-bit)
       movwf chksum
       return
```

Dado que el sensor, según la medición de que se trate, puede demorar más de 200ms en realizarla, y sabiendo que nos lo indicará poniendo el pin DATA en estado bajo, lo que haremos es interrogarlo periódicamente; esta

es la razón por la cual partimos la tarea en tres subrutinas, de modo que el programa principal puede iniciar una medición, realizar sus tareas normales e ir interrogando periódicamente al sensor, para finalmente obtener el resultado cuando éste le indica que está disponible:

```
initsn  call s_connectionreset        ; inicializa al sensor
        ; ...
        call s_measure_t              ; inicia medición de temperatura
        btfsc STATUS,C                ; verifica si hubo error (reset) o espera
        goto hubounerror
        ;...                          ; OK, empezó la medición

        call s_measure_wait           ; pregunta si ya terminó
        btfsc STATUS,C                ; verifica respuesta
        goto hayqueesperar            ; espera, verifica si ya esperó demasiado
        call s_measure_get            ; obtiene resultado
        movf value,W                  ; y lo guarda en su destino
        movwf temp
        movf value+1,W
        movwf temp+1
```

Las rutinas desarrolladas fueron escritas teniendo en cuenta las especificaciones del fabricante, para 4MHz de clock. La cantidad de calls anidadas se comprobó para un PIC16F630. Para otras aplicaciones, se sugiere al interesado revisar que se cumplan los tiempos detallados en la hoja de datos del SHT-71 y la cantidad de calls permitidos por el hardware stack del PIC utilizado.

El archivo adjunto contiene la totalidad de los listados.