Remote Temperature Sensor (RTS)

The purpose of this application is to measure temperature using the 8-pin PICmicro™ and SMT 160-30 temperature sensors. The temperature range is -45°C to +127°C. The Remote Temperature Sensor is part of a system that can monitor the temperature in 8 different points. Each RTS has its own address (0 to 7) and communicates with the master by a simple protocol. Because of the low consumption of the RTS it is powered from the data line (something like Dallas bus). For communication is used 9 bit Asynchronous serial protocol with baud rate 9600. The 9-th bit indicates that the byte is address.

The protocol is: Master sends address.
The RTS which address matches sends data
Master sends address every 100 ms. The time between communications is used by the powering capacitor to charge up, so in this time the line must be in 1.
The slave monitors the line and when it finds start bit starts receiving. If the address matches, a function MakeTemp is called. This function measures the temperature and places the result in ByteOut. Next ByteOut is sent out and the 9th bit is sent as 0 (Byte is Data).

ABOUT THE SMT 160-30

This is a temperature sensor which output is PWM signal. The temperature depends of the DC of the signal and is calculated by the formula:

$$ T = \frac{DC - 0.320}{0.0047} = \frac{DC}{0.0047} - \frac{0.320}{0.0047} = \frac{DC}{T_p} - 68 $$

Where T is °Celsius.
The temperature range is -45 to 130 °C. The frequency changes in that range from 1KHz to 4KHz.

To find the temperature, we must expand the above formula. It uses float numbers, and it will be easy if they are int.

$$ T \times 0.0047 = DC - 0.320 $$

$$ DC = \frac{T_p}{T_1} $$

where

$T_p$ is the pulse time; $T_1$ is the period.

To minimize the error from the division, it is better to expand the equation to:

$$ T = 213 \times \frac{T_p}{T_1} - 68 = (255 - 32 - 8 - 2) \times \frac{T_p}{T_1} - 68 $$

$$ T = \left( \frac{255 \times T_p}{T_1} \right) - \left( \frac{32 \times T_p}{T_1} \right) - \left( \frac{8 \times T_p}{T_1} \right) - \left( \frac{2 \times T_p}{T_1} \right) - 68 $$

It is easy to make the multiplication because the one of the multiplicands is power of 2 and it is enough to shift the other multiplicand.

Using this method of calculation the error is less than the error of the SMT 160-30.

At least the temperature is placed in reg Tmp and then in ByteOut. The type of this variables is signed char.
FIGURE 1: BLOCK DIAGRAM

```
+5V

RTS2

RTS8

Master
```

System Schematic

FIGURE 2: SCHEMATIC

```
+5V

4.7KΩ

GND

D1N4148

100μF

SMT 160-30

GND

VDD

GP4

GP2

GP0

SW1

GP1

SW2

GP3

SW3

To Master
```

Schematic of RTS Module
BILL OF MATERIALS (BOM)

<table>
<thead>
<tr>
<th>Part#</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT 160-30</td>
<td>SMARTEC</td>
</tr>
<tr>
<td>PIC12C508</td>
<td>MICROCHIP</td>
</tr>
<tr>
<td>RESISTOR 4.7K</td>
<td></td>
</tr>
<tr>
<td>DIP SW - RS04</td>
<td></td>
</tr>
<tr>
<td>D1N4148</td>
<td></td>
</tr>
<tr>
<td>C 100 μF</td>
<td></td>
</tr>
</tbody>
</table>

MICROCHIP TOOLS USED

MPASM V01.50, MPLAB 3.22
APPENDIX A:  SOURCE CODE

;****************************************************************
; Figure1.ASM
;
;****************************************************************

LIST   p=12C508

#include"inc\p12c508.inc"

__config _WDT_OFF & _IntRC_OSC & _MCLRE_OFF & _CP_OFF

RAM   equ 0x07 ;Beginning of RAM

Sensor  equ 4
InOut   equ 2

cblock RAM
  Count
  BCount
  Address
  ByteIn
  ByteOut
  Pulse
  PeriodL
  PeriodH
  ACCB0
  ACCB1
  BARGB0
  BARGB1
  LOOPCOUNT
  REMB0
  REMB1
  Tmp
  AL
  AH
endc

org 0x00

movwf OSCCAL ;calibrating the internal oscillator

clfGPIO

movlwB'00111111'
TRISGPIO

movlwB'10011111';wake up on pin change disabled
OPTION ;pullups enabled

movfGPIO,w ;reading the switches and creating
movwfAddress ;the local address

bcfAddress,2
btfscAddress,3
bsf Address,2

movlwB'00000111'
andwfAddress,f
loop

  clrf Count
loop1 ;the line must be at least 1.5 ms
      ;high, and then we must wait for
      ;a start bit
  nop
  btfss GPIO,InOut
  goto loop
  decfsz Count,f
  goto loop1

;Getting the byte from communication line to ByteIn

  btfsc GPIO,InOut;waiting for start bit
  goto $-1

  movlw .17
  movwf Count
  decfsz Count,F ;waiting about 50 us to get to
  goto $-1 ;the middle of the start bit

  movlw .8
  movwf BCount
ComIn1

  movlw .31
  movwf Count
  decfsz Count,f
  goto $-1
  rrf ByteIn,F
  bcf ByteIn,7
  btfsc GPIO,InOut
  bsf ByteIn,7
  nop
  nop
  decfsz BCount,F
  goto ComIn1

  movlw .34 ;waiting for the 9-th bit
  movwf Count
  decfsz Count,f
  goto $-1

  btfss GPIO,InOut;if this bit is 1 the received byte
  goto loop ;is adress

  movf ByteIn,w;if address matches - go on
  andlw '00000011';else goto loop
  xorwf Address,w
  btfss STATUS,2
  clrf PeriodH

  btfss GPIO,Sensor;waiting for the begining
  goto $-1 ;of the period
btfsc GPIO, Sensor;  
goto -1  

clrf TMR0  

btfss GPIO, Sensor; waiting for the end of the pulse  
goto -1  

movf TMR0, w; The value in TMR0 is the pulse width  
movwf Pulse  

Sens1  
  btfsc GPIO, Sensor  
  goto Sens2  
  
  movf TMR0, w  
  btfsc STATUS, Z  
  incf PeriodH, f  
  goto Sens1  

Sens2  
  movf TMR0, w; The value in the PeriodL and PeriodH  
  movwf PeriodL; is the Period.  
  
  call MakeTemp  
  movwf ByteOut  

; Outputs the byte placed in OutByte  
  
  movlw B'00111011'; change InOut pin to output  
  TRIS GPIO  
  
  bcf GPIO, InOut  
  
  movlw .8  
  movwf BCount  

ComOut1  
  movlw .31  
  movwf Count  
  decfsz Count, f  
  goto -1  
  
  btfsz ByteOut, 0  
  bsf GPIO, InOut  
  btfsz ByteOut, 0  
  bcf GPIO, InOut  
  
  rrf ByteOut, F  
  
  decfsz BCount, f  
  goto ComOut1  
  
  movlw .31  
  movwf Count  
  decfsz Count, f  
  goto -1  
  
  bcf GPIO, InOut; clears the 9-th bit  
  
  movlw .31  
  movwf Count  
  decfsz Count, f  
  goto -1  
  
  bcf GPIO, InOut; clears the 9-th bit
decszCount,f
goto$-1

movlw'b00111111';change InOut pin to input
TRISGPIO
gotoloop

;***************************************************************
;
; char Div(char ACCB0,char ACCB1,char BARGB0,char BARGB1);
;
; devides the 16 bit value in ACCB to the 16 bit value in BARGB
; the result is placed in the W register

Div

c1rfREMB0
clrfREMB1
movlw16
movwfLOOPCOUNT

LOOPU1616:
    RLF ACCB0,W
    RLF REMB1, F
    RLF REMB0, F
    MOVFBARGB1,W
    SUBWFREMB1, F
    MOVFBARGB0,W
    BTFS3,0
    INCFSZBARGB0,W
    SUBWFREMB0, F
    BTFS3,0
    GOTOUOK66LL
    MOVFBARGB1,W
    ADDWFREMB1, F
    MOVFBARGB0,W
    BTFS3,0
    INCFSZBARGB0,W
    ADDWFREMB0, F
    BCF 3,0

UOK66LL:
    RLF ACCB1, F
    RLF ACCB0, F
    DECSZLOOPCOUNT, F
    GOTOLOOPU1616

    movfACCB0,w
    return

;***************************************************************
;
; char MakeTemp(char T1, char TcL, char TcH)
;
; input: Pulse in T1, Period in Tc;
; output: calculated temperature in W register
; calculates the temperature by the formula:
;
; \[ T_1 \times 256/T_c - T_1 \times 32/T_c - T_1 \times 8/T_c - T_1 \times 2/T_c - 68 \]
;
MakeTemp

; \[ Tmp = \text{Div}(0, T_1, T_{cL}, T_{cH}) \]
clr ACCB0
movf Pulse, w
movwf ACCB1
movf PeriodL, w
movwf BARGB0
movf PeriodH, w
movwf BARGB1

callDiv
movwf Tmp

clr AL
movf Pulse, w
movwf AH

bcf STATUS, C
rrf AH, F
rrf AL, F
rrf AH, F
rrf AL, F
rrf AH, F
rrf AL, F

; \[ Tmp -= \text{Div}(AL, AH, T_{cL}, T_{cH}) \]

movf AL, w
movwf ACCB0
movf AH, w
movwf ACCB1
movf PeriodL, w
movwf BARGB0
movf PeriodH, w
movwf BARGB1

callDiv

subwf Tmp, f

clr AH
movf Pulse, w
movf AL

bcf 3, 0
rlf AH, F
rlf AL, F
rlf AH, F
rlf AL, F
rlf AH, F

; \[ Tmp -= \text{Div}(AL, AH, T_{cL}, T_{cH}) \]

movf AL, w
movwf ACCB0
movf AH, w
movwf ACCB1
movf PeriodL, w
movwf BARGB0
movf PeriodH, w
movwf BARGB1

callDiv

subwf Tmp, f

clrfr
movf Pulse, w
movwf AL

bcf 3, 0
rlf AL, F
rlf AH, F

; Tmp = Div(AL, AH, TcL, TcH);

movf AL, w
movwf ACCB0
movf AH, w
movwf ACCB1
movf PeriodL, w
movwf BARGB0
movf PeriodH, w
movwf BARGB1

callDiv

subwf Tmp, f

; Tmp = 68;

movlw .68
subwf Tmp, w

return

end