Remote Temperature Sensor (RTS)

Author: *Slav E. Slavov*
Ell
Sliven, Bulgaria
Email: slav_slavov@hotmail.com

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:

$$DC = 0.320 + 0.0047 * T$$

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.

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

$$T = \frac{DC - 0.320}{0.0047} = \frac{DC}{0.0047} = \frac{0.320}{0.0047} = 213 \times DC - 68$$

$$DC = \frac{Tp}{T1}$$

where

Tp is the pulse time; T1 is the period.

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

$$T = 213 \times \frac{Tp}{T1} - 68 = (255 - 32 - 8 - 2) \times \frac{Tp}{T1} - 68$$

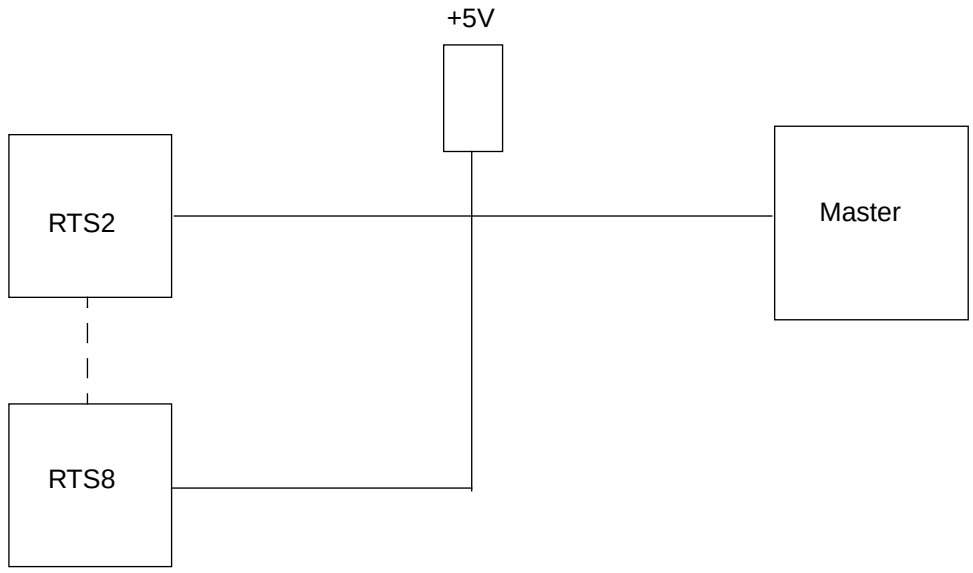
$$T = \frac{(255 * Tp)}{T1} - \frac{(32 * Tp)}{T1} - \frac{(8 * Tp)}{T1} - \frac{(2 * Tp)}{T1} - 68$$

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.

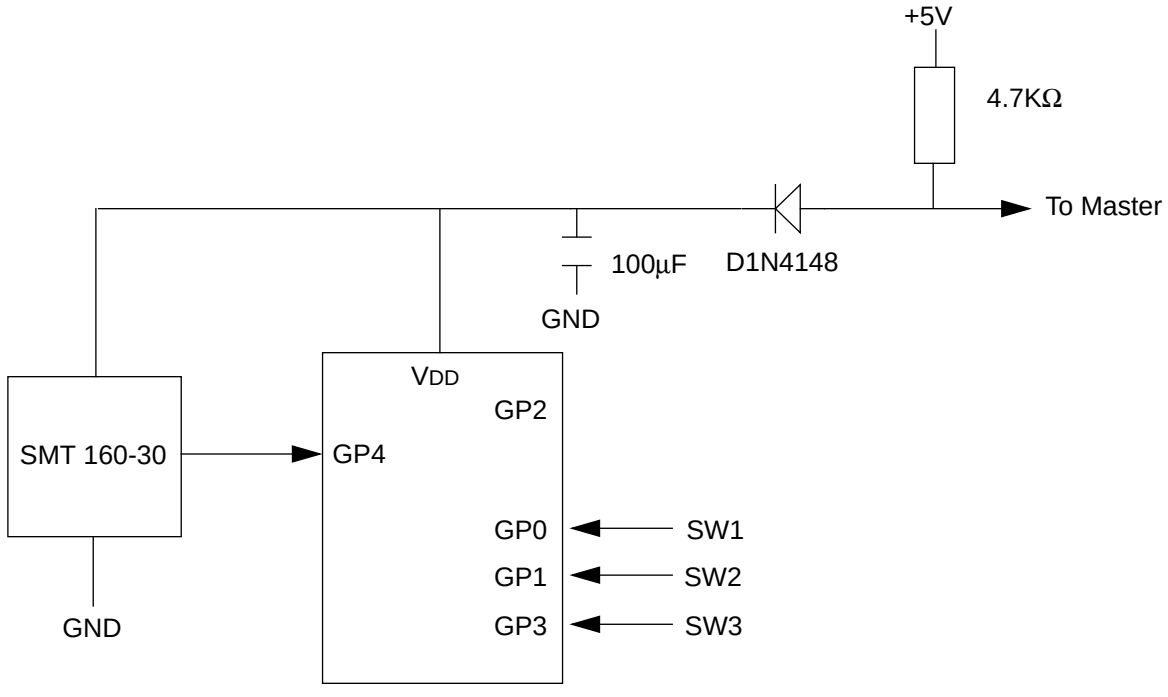
At least the temperature is placed in `reg Tmp` and then in `ByteOut`. The type of this variables is `signed char`.

FIGURE 1: BLOCK DIAGRAM



System Schematic

FIGURE 2: SCHEMATIC



Schematic of RTS Module

BILL OF MATERIALS (BOM)

Part#	Manufacturer
SMT 160-30	SMARTEC
PIC12C508	MICROCHIP
RESISTOR 4.7K	
DIP SW - RS04	
D1N4148	
C 100 µF	

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    ;Begining of RAM

Sensor  equ 4
InOut   equ 2


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

    org 0x00

    movwfOSCCAL;calibrating the internal oscillator

    clrfGPIO



    movlwB'00111111'
    TRISGPIO

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


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

    bcfAddress,2
    btfscAddress,3
    bsf Address,2

    movlwB'00000111'
    andwfAddress,f
```

```
loop

    clrfCount
loop1               ;the line must be at least 1.5 ms
    nop             ;high, and then we must wait for
    btfssGPIO,InOut;a start bit
    gotoloop
    decfszCount,f
    gotoloop1


;Getting the byte from communication line to ByteIn

    btfscGPIO,InOut;waiting for start bit
    goto$-1


    movlw.17
    movwfCount

    decfszCount,F  ;waiting about 50 us to get to
    goto$-1        ;the middle of the start bit


    movlw.8
    movwfBCount

ComIn1
    movlw.31
    movwfCount

    decfszCount,f
    goto$-1

    rrf ByteIn,F

    bcf ByteIn,7
    btfscGPIO,InOut
    bsf ByteIn,7
    nop
    nop
    decfszBCount,F
    gotoComIn1


    movlw.34       ;waiting for the 9-th bit
    movwfCount

    decfszCount,f
    goto$-1


    btfssGPIO,InOut;if this bit is 1 the received byte
    gotoloop       ;is adress


    movfByteIn,w;if address matches - go on
    andlwB'00000111';else goto loop
    xorwfAddress,w
    btfssSTATUS,Z

    clrfPeriodH

    btfssGPIO,Sensor;waiting for the begining
    goto$-1        ;of the period
```

```
    btfscGPIO,Sensor;
    goto$-1        ;

    clrfTMR0

    btfssGPIO,Sensor;waiting for the end of the pulse
    goto$-1

    movfTMR0,w     ;The value in TMR0 is the pulse width
    movwfPulse

Sens1
    btfscGPIO,Sensor
    gotoSens2

    movfTMR0,w
    btfscSTATUS,Z
    incfPeriodH,f
    gotoSens1
Sens2
    movfTMR0,w     ;The value in the PeriodL and PeriodH
    movwfPeriodL   ;is the Period.

    callMakeTemp
    movwfByteOut


;Outputs the byte placed in OutByte

    movlwB'00111011';change InOut pin to output
    TRISGPIO

    bcf GPIO,InOut

    movlw.8
    movwfBCount

ComOut1
    movlw.31
    movwfCount

    decfszCount,f
    goto$-1

    btfscByteOut,0
    bsf GPIO,InOut
    btfssByteOut,0
    bcf GPIO,InOut

    rrf ByteOut,F

    decfszBCount,f
    gotoComOut1


    movlw.31
    movwfCount

    decfszCount,f
    goto$-1

    bcf GPIO,InOut;clears the 9-th bit

    movlw.31
    movwfCount
```

```
    decfszCount,f
    goto$-1


    movlwB'00111111';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

    clrfREMB0
    clrfREMB1
    movlw16
    movwfLOOPCOUNT

LOOPU1616:
    RLF ACCB0,W
    RLF REMB1, F
    RLF REMB0, F
    MOVFBARGB1,W
    SUBWFREMB1, F
    MOVFBARGB0,W
    BTFSS3,0
    INCFSZBARGB0,W
    SUBWFREMB0, F

    BTFSC3,0
    GOTOUOK66LL
    MOVFBARGB1,W
    ADDWFREMB1, F
    MOVFBARGB0,W
    BTFSC3,0
    INCFSZBARGB0,W
    ADDWFREMB0, F

    BCF 3,0

UOK66LL:
    RLF ACCB1, F
    RLF ACCB0, F

    DECFSZLOOPCOUNT, 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:
;
;  T1*256/Tc - T1*32/Tc - T1*8/Tc - T1*2/Tc - 68
;


MakeTemp

;  Tmp=Div(0,T1,TcL,TcH);
    clrfACCB0
    movfPulse,w
    movwfACCB1
    movfPeriodL,w
    movwfBARGB0
    movfPeriodH,w
    movwfBARGB1

    callDiv
    movwfTmp

    clrfAL
    movfPulse,w
    movwfAH

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


  ;Tmp-=Div(AL,AH,Tcl,TcH)

    movfAL,w
    movwfACCB0
    movfAH,w
    movwfACCB1
    movfPeriodL,w
    movwfBARGB0
    movfPeriodH,w
    movwfBARGB1

    callDiv

    subwfTmp,f

    clrfAH
    movfPulse,w
    movwfAL

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

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

    movfAL,w
    movwfACCB0
    movfAH,w
    movwfACCB1
```

```
    movfPeriodL,w
    movwfBARGB0
    movfPeriodH,w
    movwfBARGB1

    callDiv

    subwfTmp,f

    clrfAH
    movfPulse,w
    movwfAL

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

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

    movfAL,w
    movwfACCB0
    movfAH,w
    movwfACCB1
    movfPeriodL,w
    movwfBARGB0
    movfPeriodH,w
    movwfBARGB1

    callDiv

    subwfTmp,f


  ;Tmp-=68;

    movlw.68
    subwfTmp,w

    return

    end
```

NOTES: